UNIVERSITY OF CALIFORNIA, MERCED

Data-Driven Visual Synthesis for Natural Image and Video Editing

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering and Computer Science

by

Yijun Li

Committee in charge:

Professor Ming-Hsuan Yang, Chair
Professor Shawn Newsam
Professor Sungjin Im
Doctor Chen Fang

2019

The dissertation of Yijun Li is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Professor Shawn Newsam

Professor Sungjin Im

Doctor Chen Fang

Professor Ming-Hsuan Yang Chair

University of California, Merced

2019

To my parents for their love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I still kept the email received four years ago from my advisor Ming-Hsuan Yang, officially talking about providing me with the offer to pursue the Ph.D. degree at UC Merced. Now with the mission completed, my gut feeling is that I live up to expectations. Luckily I manage to grasp this great opportunity and fulfill it with fruitful outputs. Such a long journey is full of bitter and joy, which will be deeply engraved on my memory forever.

I would like to first thank Ming-Hsuan for his guidance, passion, and support throughout my years at Merced. First, he creates the free research atmosphere in the lab. Everybody is able to select their most interested vision-related topic and then collaborate with him. Ming-Hsuan might not be expert in every research topic but his intuitions and rich experiences help me quickly be on the right direction towards the goal. Second, he holds high standard for the research work. He often serves as a strict reviewer to question each part of the work until I address all of them. Third, he is responsible for every project that he is involved in. It is hard to imagine how busy his schedule is, but he seldom misses weekly meetings, in order to know the progress and provide immediate help. When it comes to the writing, I do not remember how many times my poor writing has made him mad (sorry about that) but he keeps going over every sentence to teach me how to write the professional academic article. All those gradually turns me to be a qualified researcher.

I would also like to thank Jia-Bin Huang from Virginia Tech who works closely with me to publish my first top conference paper and the first top journal paper. When I just arrived at Merced, Jia-Bin helps set up the whole project for me to start. By working with him for nearly one year, I learned the full process of doing a project, from zero to the final publication. Especially when our submission was first rejected, Jia-Bin taught me how to face the rejection in the right way and then overcome those setbacks. Without his help of my first project, it is impossible for me to succeed in the following projects. I feel super grateful and sincerely wish him the best of luck to get the position of tenured professor.

In addition to the research life in lab, another big chapter of my research happened over summers when I did the internships. I met so many great mentors from industry.

First and foremost, I would like to thank mentors from Adobe Research as I was interning there three times. They are Chen Fang, Eli Shechtman, Jimei Yang, Aaron Hertzmann, Zhaowen Wang, and Xin Lu. We published four top papers together and experience lots of great moments. Chen is my first mentor and often shares his working experiences with me, which is later demonstrated to be very helpful when I start the job hunting. Eli shows me how to balance the research and life. I will never forget that he pulled me into the water during the drift trip at Seattle. Aaron's early-planning way of working style and even higher (than my advisor) standard for the research work impresses me most and help shape my attitude when doing the research. All the help from mentors contribute to my fruitful outputs at Adobe and certainly help me win the Adobe Fellowship (2018). Also big thanks to Adobe. I really love working here with so many talented people.

Interning at Nvidia Research is also a great experience. I would like to thank my mentors Ming-Yu Liu and Jan Kautz. We did a fantastic project about photo stylization there. Until today it is still my most popular project on Github, with nearly 10K stars. That is the first time when I feel how a research project could impact and facilitate so many people's creations. Ming-Yu is a very responsible mentor who keeps thinking about my project and stops by to share ideas with me everyday. He also pays attention to every detail of the project, such as redesigning the framework figure in the paper by himself. I still remember when he was holding his five-day daughter in one arm and helped me revise the paper with another hand to chase the CVPR deadline.

The last part of thanks goes to my parents and my girlfriend Chuhang Zou. My parents are both Ph.D.s, which is the biggest inspiration for me to start this journey. Thanks for their unbroken chain of supports and love from thousands of miles away. I missed all the spring festivals in those four years which makes me feel guilty, but I know they are super proud of me. My girlfriend Chuhang is also a Ph.D. who fully understands what I am doing everyday. Though we are in long distance relationship, she never complains and keeps supporting me to go through all the up and down moments. Words are not enough to express how thankful I feel. I love them forever.

VITA

2012	B. S. in Control Science and Engineering, Zhejiang University, Hangzhou
2015	M. S. in Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai
2019	Ph. D. in Electrical Engineering and Computer Science, University of California, Merced

PUBLICATIONS

Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang, *Im2Pencil: Controllable Pencil Illustration from Photographs*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang, *Joint Image Filtering with Deep Convolutional Networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018, DOI: 10.1109/TPAMI.2018.2890623.

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang, *Flow-Grounded Spatial-Temporal Video Prediction from Still Images*, European Conference on Computer Vision (ECCV), 2018.

Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz, *A Closed-form Solution to Photorealistic Image Stylization*, European Conference on Computer Vision (ECCV), 2018.

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang, *Universal Style Transfer via Feature Transforms*, Advances in Neural Information Processing Systems (NIPS), 2017.

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang, *Diversified Texture Synthesis with Feed-forward Networks*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (**Spotlight**)

Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang, *Generative Face Completion*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang, *Deep Joint Image Filtering*, European Conference on Computer Vision (ECCV), 2016.

ABSTRACT OF THE DISSERTATION

Data-Driven Visual Synthesis for Natural Image and Video Editing

by

Yijun Li
Doctor of Philosophy in Electrical Engineering and Computer Science
University of California Merced, 2019
Professor Ming-Hsuan Yang, Chair

Visual data are what make our daily life fun. Often times, we consume those data created by experts in related fields, e.g., appreciating artworks drawn by famous painters or watching movies shot by professional directors. How about creating the desired data that show our own feelings, ideas and creativity by ourselves? This comes to the *Visual Synthesis*, which is the process of synthesizing new data or altering existing data. However, attempts from large amounts of non-experts often end up deviating from the manifold of real natural data, leading to unrealistic results with undesired artifacts. The goal of all research work in this thesis is to develop effective computational models to preserve visual realism and facilitate more stunning creations. We mainly develop data-driven approaches by learning from large amounts of existing created visual data and explore effective models so that they can generalize to enormous unseen target data. Essentially, visual synthesis is working on manipulating different factors that form the final observed data, such as structure, style, content, motion and so on. Along this direction, we mainly explore four synthesis tasks for various image and video editing scenarios, including structure enhancement, style transfer, content filling and motion prediction.

Chapter 3 describes a joint filtering method on enhancing the sharpness of low-quality **structures** in images. Chapter 4 presents how to alter the **style** of an image with another

brand new style. We propose a universal style transfer algorithm that works for arbitrary style inputs. Chapter 5 focuses on how to fill in the missing **content** in images in order to remove occlusions. We aim at the face completion which is more challenging as it often requires generating semantically new pixels for the missing key components. In Chapter 6, we present a novel algorithm on how to generate pixel-level future frames in multiple time steps given one still image. This represents an important step towards simulating similar *preplay* activities that might constitute an automatic prediction mechanism in human visual cortex.

Chapter 1

Introduction

1.1 Overview

The visual data (e.g., images, videos) are what make make our daily life fun. Without them, we would see nothing but a news feed full of text. The old saying "A picture is worth a thousand words" also demonstrates that a complex idea can be conveyed with just a single picture which conveys its meaning or essence more effectively than a description does. Therefore instead of passively watching created data by experts, each one of non-expert users actively has the desire to manipulate and edit their own data based on different preferences and requirements. This makes an effective computation model in high demand to simplify the user's editing process. As shown in Figure 1.1, the computer generated results should agree with the user's intention. A deeper interpretation is that the editing results should not deviate from the manifold of real natural data, which are expected to be as realistic as possible without any artifacts. This motivates us to develop powerful editing tools to enable everyone's creation.

In essential, visual synthesis focuses on manipulating different factors that form an image. Figure 1.2 shows an example of real editing case. Given an target input image in (a), the user may not care about detailed local **structures** (e.g., those on sky and sea)

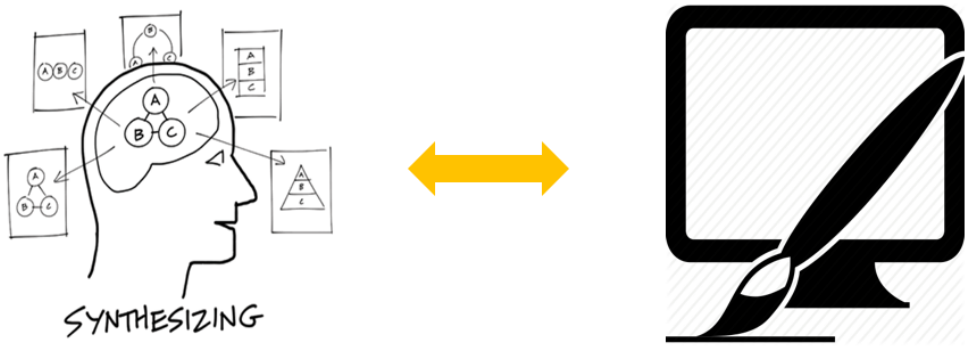
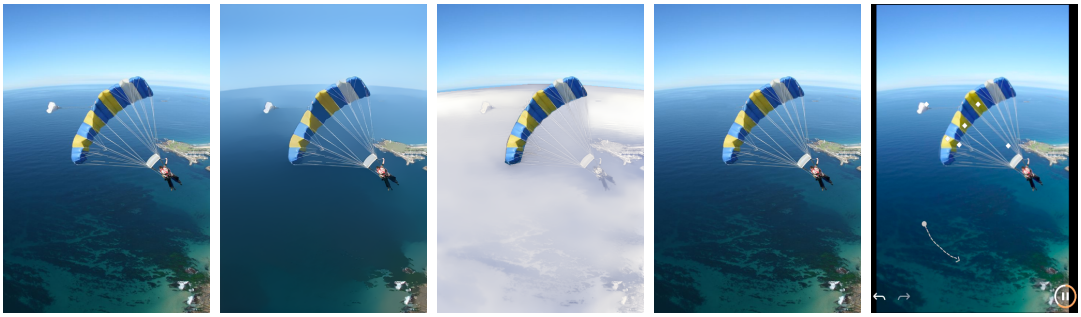


Figure 1.1: We target on developing effective computational models to enable the visual synthesis that faithfully reflects people's (especially for non-expert users) ideas in mind.



(a) Target (b) Structure (c) Style (d) Content (e) Motion

Figure 1.2: Manipulating different factors of an image in visual synthesis.

but lines of the parachute are important structures to preserve. Thus an image smoothing model should come up with a kind of selective smoothing ability to help users achieve the goal in (b). Another user may want to change the **style** of the background so that the pilot looks like flying over the arctic regions. We provide an image stylization tool to stylize the sea with the snow-like feel in (c). In addition, there is a undesirable white small **content** on left and we automatically remove it in (d) with a seamless result. Finally, to turn the image to a vivid GIF version, the user may want the sea to have a flowing-like **motion** by simply indicating a directional path in (e). Will this also be possible?

Based on such a real editing case, we accordingly explore four synthesis tasks in this thesis, including structure enhancement, style transfer, content filling and motion prediction. Those four factors, i.e., the structure, style, content and motion, are among the most basic and important factors that constitute the final observed data. We mainly develop data-driven approaches by learning from large amounts of existing created visual data and explore effective models so that they can generalize to enormous unseen target data.

1.2 Dissertation Outline

In Chapter 2, we conduct a thorough literature review that relates to the visual synthesis on different factors and beyond. Chapter 3-6 describe the main technical details, experimental results and in-depth analysis on how to manipulate four factors, i.e., the structure, style, content and motion.

More specifically, in Chapter 3, we propose a learning-based approach for constructing joint filters based on Convolutional Neural Networks. The proposed algorithm can selectively transfer salient structures that are consistent with both the guidance and target images. In Chapter 4, we present a universal style transfer algorithm that does not require learning for each individual style. We integrate the whitening and coloring transforms in the feed-forward passes to match the statistical distributions and correlations between content and style. In Chapter 5, we propose an effective face completion algorithm using a deep generative model that incorporates the adversarial learning. We demonstrate that our model is able to deal with a large area of missing pixels in arbitrary shapes and generate realistic face completion results. In Chapter 6, we study the problem of generating consecutive multiple future frames and formulate the multi-frame prediction task as a multiple time step flow (multi-flow) prediction phase followed by a flow-to-frame synthesis phase. We conduct extensive experiments on different types of motion to validate its effectiveness.

We summarize the contributions and highlight three future directions to explore in

Chapter 7.

Chapter 2

Literature Review

2.1 Structure Enhancement

The structure enhancement is the procedure of improving the sharpness of structures in low-quality target images. Joint image filtering is a typical task for enhancement in the field of computer vision. The basic idea is to leverage a reference image as a prior and transfer the structural information to the target image for image enhancement. Those filters can be categorized into two main classes based on explicit filter construction or global optimization of data fidelity and regularization terms.

Explicit joint filters compute the filtered output as a weighted average of neighboring pixels in the target image. The bilateral filters [110, 140, 61, 74, 148, 5] and guided filters [46] are representative algorithms in this class. The filter weights, however, depend solely on the local structure of the guidance image. Therefore, erroneous or extraneous structures may be transferred to the target image due to the lack of consistency constraints. In contrast, our model considers the contents of both images based on feature maps and enforces consistency implicitly through learning from examples.

Numerous approaches formulate joint filtering based on a global optimization framework. The objective function typically consists of two terms: data fidelity and regulariza-

tion terms. The data fidelity term ensures that the filtering output is close to the input target image. These techniques differ from each other mainly in the regularization term that encourages the output to have a similar structure with the guidance image. The regularization term can be defined according to texture derivatives [20], mid-level representations [88] such as segmentation and saliency, filtering outputs [44], or mutual structures shared by the target and guidance image [100]. However, global optimization based methods rely on hand-designed objective functions that may not reflect the complexities of natural images. Furthermore, these approaches involve iterative optimization and are often time-consuming. In contrast, our method learns how to selectively transfer important details directly from the RGB/depth data. Although the training process is time-consuming, the learned model is efficient during run-time.

Learning-based image filters. With significant success in high-level vision tasks [63], substantial efforts have been made to construct image filters using learning algorithms and CNNs. For example, the conventional bilateral filter can be improved by replacing the predefined filter weights with those learned from a large amount of data [55, 39]. In the context of joint depth upsampling, Tai et al. [54] use a multi-scale guidance strategy to improve upsampling performance. Gu et al. [43] adjust the original guidance dynamically to account for the iterative updates of the filtering results. However, these methods [54, 43] are limited to the application of depth map upsampling. In contrast, our goal is to construct a generic joint filter for various applications using target/guidance image pairs in different visual domains.

Deep models for low-level vision. In addition to filtering, deep learning models have also been applied to other low-level vision and computational photography tasks. Examples include image denoising [9], raindrop removal [28], image super-resolution [23], image deblurring [147] and optical flow estimation [91]. Existing deep learning models for low-level vision use either one input image [23, 9, 28, 134] or two images in the same domain [91]. In contrast, our network can accommodate two streams of inputs by *heterogeneous*

domains, e.g., RGB/NIR, flash/non-flash, RGD/Depth, intensity/color. Our network architecture bears some resemblance to that in Dosovitskiy et al. [91]. The main difference is that the merging layer used in [91] is a correlation operator while our model integrates the inputs through concatenating the feature responses. Furthermore, we adopt residual learning by introducing skip connections.

2.2 Style Transfer

Style transfer is the technique of recomposing a content image in the style of another style image for image synthesis. Existing style transfer methods are mostly example-based [48, 104, 103, 33, 72]. The image analogy method [48] aims to determine the relationship between a pair of images and then apply it to stylize other images. As it is based on finding dense correspondence, analogy-based approaches [104, 103, 33, 72] often require that a pair of image depicts the same type of scene. Therefore these methods do not scale to the setting of arbitrary style images well.

Recently, Gatys et al. [35, 36] proposed an algorithm for arbitrary stylization based on matching the correlations (Gram matrix) between deep features extracted by a trained network classifier within an iterative optimization framework. Numerous methods have since been developed to address different aspects including speed [114, 70, 56], quality [115, 69, 130, 129], user control [37], diversity [116, 141], semantics understanding [33, 11] and photorealism [80]. It is worth mentioning that one of the major drawbacks of [35, 36] is the inefficiency due to the optimization process. The improvement of efficiency in [114, 70, 56] is realized by formulating the stylization as learning a feed-forward image transformation network. However, these methods are limited by the requirement of training one network per style due to the lack of generalization in network design.

Most recently, a number of methods have been proposed to empower a single network to transfer multiple styles, including a model that conditioned on binary selection units [141], a network that learns a set of new filters for every new style [13], and a novel

conditional normalization layer that learns normalization parameters for each style [27]. To achieve arbitrary style transfer, Chen et al. [14] first propose to swap the content feature with the closest style feature locally. Meanwhile, inspired by [27], two following work [127, 40] turn to learn a general mapping from the style image to style parameters. One closest related work [53] directly adjusts the content feature to match the mean and variance of the style feature. However, the generalization ability of the learned models on unseen styles is still limited.

Different from the existing methods, our approach performs style transfer efficiently in a feed-forward manner while achieving generalization and visual quality on arbitrary styles. Our approach is closely related to [53], where content feature in a particular (higher) layer is adaptively instance normalized by the mean and variance of style feature. This step can be viewed as a sub-optimal approximation of the WCT operation, thereby leading to less effective results on both training and unseen styles. Moreover, our encoder-decoder network is trained solely based on image reconstruction, while [53] requires learning such a module particularly for stylization task. We evaluate the proposed algorithm with existing approaches extensively on both style transfer and texture synthesis tasks and present in-depth analysis.

2.3 Content Filling

Content filling, also known as completion or inpainting, refers to fill the missing or masked regions in images with plausibly synthesized contents. It has been studied in numerous contexts, e.g., inpainting, texture synthesis, and sparse signal recovery. Since a thorough literature review is beyond the scope of this work, and we discuss the most representative methods to put our work in proper context.

An early inpainting method [7] exploits a diffusion equation to iteratively propagate low-level features from known regions to unknown areas along the mask boundaries. While it performs well on inpainting, it is limited to deal with small and homogeneous

regions. Another method has been developed to further improve inpainting results by introducing texture synthesis [8]. In [153], the patch prior is learned to restore images with missing pixels. Recently Ren et al. [97] learn a convolutional network for inpainting. The performance of image completion is significantly improved by an efficient patch matching algorithm [4] for nonparametric texture synthesis. While it performs well when similar patches can be found, it is likely to fail when the source image does not contain sufficient amount of data to fill in the unknown regions. We note this typically occurs in object completion as each part is likely to be unique and no plausible patches for the missing region can be found. Although this problem can be alleviated by using an external database [45], the ensuing issue is the need to learn high-level representation of one specific object class for patch match.

Wright et al. [131] cast image completion as the task for recovering sparse signals from inputs. By solving a sparse linear system, an image can be recovered from some corrupted input. However, this algorithm requires the images to be highly-structured (i.e., data points are assumed to lie in a low-dimensional subspace), e.g., well-aligned face images. In contrast, our algorithm performs completion without strict constraints.

Image generation. Vincent et al. [120] introduce denoising autoencoders that learn to reconstruct clean signals from corrupted inputs. In [26], Dosovitskiy et al. demonstrate that an object image can be reconstructed by inverting deep convolutional network features (e.g., VGG [105]) through a decoder network. Kingma et al. [59] propose variational autoencoders (VAEs) which regularize encoders by imposing prior over the latent units such that images can be generated by sampling from or interpolating latent units. However, the generated images by a VAE are usually blurry due to its training objective based on pixel-wise Gaussian likelihood. Larsen et al. [65] improve a VAE by adding a discriminator for adversarial training which stems from the generative adversarial networks (GANs) [42] and demonstrate more realistic images can be generated.

The most recent work proposed by Deepak et al. [89] applies an autoencoder and in-

tegrates learning visual representations with image completion. However, this approach emphasizes more on unsupervised learning of representations than image completion. In essence, this is a chicken-and-egg problem. Despite the promising results on object detection, it is still not entirely clear if image completion can provide sufficient supervision signals for learning high-level features. On the other hand, semantic labels or segmentations are likely to be useful for improving the completion results, especially on a certain object category. With the goal of achieving high-quality image completion, we propose to use an additional semantic parsing network to regularize the generative networks. Our model deals with severe image corruption (large region with missing pixels), and develops a combined reconstruction, adversarial and parsing loss for face completion.

2.4 Motion Prediction

Action prediction. The macroscopic analysis of prediction based on the given frame(s) can be predicting what event is going to happen [144, 64, 50], trajectory paths [60], or recognizing the type of human activities [122, 126]. Some of early methods are supervised, requiring labels (e.g., bounding boxes) of the moving object. Later approaches [126] realize the unsupervised way of prediction by relying on the context of scenes. However, these approaches usually only provide coarse predictions of how the future will evolve and are unable to tell richer information except for an action (or event) label.

Pixel-level frame prediction. Recent prediction methods move to the microcosmic analysis of more detailed information in the future. This is directly reflected by requiring the pixel-level generation of future frames in multiple time steps. With the development of deep neural networks, especially when recursive modules are extensively used, predicting realistic future frames has dominated. Much progress has been made in the generated quality of future outputs by designing different network structures [109, 87, 83] or using different learning techniques, including adversarial loss [123, 71], motion/content

separation [118, 113, 19], and transformation parameters [32, 124].

Our work also aims at accurate frame predictions but the specific setting is to model the uncertainties of multi-frame prediction given a single still image as input. In terms of multi-frame predictions on still images, the closest work to ours are [12, 119]. However, [12] only predicts the pose information and the proposed model is deterministic. The work in [119] also estimates pose first and then uses an image-analogy strategy to generate frames. But their pose generation step relies on observing multiple frames. Moreover, both approaches employ the recursive module (e.g., recurrent neural networks) for consecutive predictions which may overemphasize on learning the temporal information only. Instead, we use the 3D convolutional layer [111] which takes a volume as input. Since both spatial and temporal information are encoded together, the 3D convolution can generally capture correlations between the spatial and temporal dimension of signals, thereby rendering distinctive spatial-temporal features [111]. In addition, both [12, 119] focus on human dynamics while our work targets on both articulated objects and dynamic textures.

In terms of modeling future uncertainties, two methods [136, 125] are closely related. However, Xue *et al.* [136] only model the uncertainty in the next one-step prediction. If we iteratively run the one-step prediction model for multi-step predictions, the frame quality will degrade fast through error accumulations, due to the lack of temporal relationships modeling between frames. Though Walker *et al.* [125] could keep forecasting over the course of one second, instead of predicting real future frames, it only predicts the dense trajectory of pixels. Also such a trajectory-supervised modeling requires laborious human labeling. Different from these methods, our approach integrates the multi-frame prediction and uncertainty modeling in one model.

Chapter 3

Deep Joint Image Filtering

Joint image filters can leverage the guidance image as a prior and transfer the structural details from the guidance image to the target image for suppressing noise or enhancing spatial resolution. Existing methods rely on various kinds of explicit filter construction or hand-designed objective functions. It is thus difficult to understand, improve, and accelerate them in a coherent framework. In this work, we propose a learning-based approach to construct a joint filter based on Convolutional Neural Networks. In contrast to existing methods that consider only the guidance image, our method can selectively transfer salient structures that are consistent in both the guidance and target images. We show that the model trained on a certain type of data, e.g., RGB and depth images, generalizes well for other modalities, e.g., Flash/Non-Flash and RGB/NIR images. We validate the effectiveness of the proposed joint filter through extensive comparisons with state-of-the-art methods.

3.1 Introduction

Image filtering with guidance signals, known as *joint* or *guided filtering*, has been successfully applied to numerous computer vision and computer graphics tasks, such as depth map enhancement [140, 88, 31], joint upsampling [61, 140], cross-modality noise reduction [137, 46, 100], and structure-texture separation [135, 148]. The wide applicability of joint filters can be attributed to the adaptability in handling visual signals in various image domains and modalities, as shown in Figure 3.1. For a target image, the guidance image can either be the target image itself [110, 46], high-resolution RGB images [46, 88, 31], images from different sensing modalities [29, 38, 137], or filter outputs from previous iterations [148]. The basic idea behind joint image filtering is that we can transfer the important structural details contained in the guidance image to the target image. The main goal of joint filtering is to enhance the degraded target image due to noise or low spatial resolution while avoiding transferring extraneous structures that do not originally exist in the target image, e.g., texture-copying artifacts.

Several approaches have been developed to transfer structures in the guidance image to the target image. One category of algorithms is to construct joint filters for specific tasks. For example, the bilateral filtering algorithm [110] constructs spatially-varying filters that reflect local image structures (e.g., smooth regions, edges, textures) in the guidance image. Such filters can then be applied to the target image for edge-aware smoothing [110] or joint upsampling [61]. On the other hand, the guided image filter [46] assumes a locally linear model over the guidance image for filtering. However, these filters share one common drawback. That is, the filter construction considers only the information contained in the guidance image and remains fixed (i.e., static guidance). When the local structures in the guidance and target images are not consistent, these methods may transfer incorrect or extraneous contents to the target image.

To address this issue, recent efforts focus on considering the common structures existing in both the target and guidance images [148, 44, 100]. These frameworks typically

Depth map upsampling	Saliency map upsampling	Chromaticity map upsampling	Flash/Non-Flash noise reduction

Figure 3.1: Sample applications of joint image filtering. The target/guidance pair (top) can be various types of cross-modality visual data. With the help of the guidance image, important structures can be transferred to the degraded target image to help enhance the spatial resolution or suppress noises (bottom). The guidance image can either be high-resolution RGB images or images from different sensing modalities.

build on iterative methods for minimizing global objective functions. The guidance signals are updated at each iteration (i.e., dynamic guidance) towards preserving the mutually consistent structures while suppressing contents that are not commonly shared in both images. However, these global optimization based methods often use hand-crafted objective functions that may not reflect natural image priors well and typically require a heavy computational load.

In this work, we propose a learning-based joint filter based on Convolutional Neural Networks (CNNs). We propose a network architecture that consists of three sub-networks and a skip connection, as shown in Figure 3.2. The first two sub-networks $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ extract informative features from both the target and guidance images. These feature responses are then concatenated as inputs for the network $\mathrm{CNN_F}$ to selectively transfer

common structures. As the target input and output images are largely similar, we introduce a skip connection, together with the output of $\mathrm{CNN_F}$ to reconstruct the filtered output. In other words, we enforce the network to focus on learning the residuals between the degraded target and the ground truth images. We train the network using large quantities of RGB/depth data and learn all the network parameters simultaneously without stage-wise training.

Our algorithm differs from existing methods in that the proposed joint image filter is purely data-driven. This allows the network to handle complicated scenarios that may be difficult to capture through hand-crafted objective functions. While the network is trained using the RGB/depth data, the network learns how to selectively transfer structures by leveraging the prior from the guidance image, rather than predicting specific values. As a result, the learned network generalizes well for handling images in various domains and modalities.

We make the following contributions in this work:

- We propose a learning-based framework for constructing a generic joint image filter. Our network takes both the target and guidance images into consideration and naturally handles the inconsistent structure problem. Using the learned joint image filter for depth upsampling, we demonstrate the state-of-the-art performance on the NYU v2 [85] and SUN RGB-D [108] dataset and achieve competitive performance on the Middlebury dataset [98, 49].

- We show that the model trained on a certain type of data (e.g., RGB/depth) generalizes well to handle image data in a variety of domains.

A preliminary version of this work was presented earlier in [143]. In this work, we significantly extend our work and summarize the main differences as follows. First, we propose an improved network architecture for joint image filtering. Instead of directly predicting filtered pixel values (as in [143]), we predict a residual image by adding a skip connection from the input target image to the output (Figure 3.2). As the residual learn-

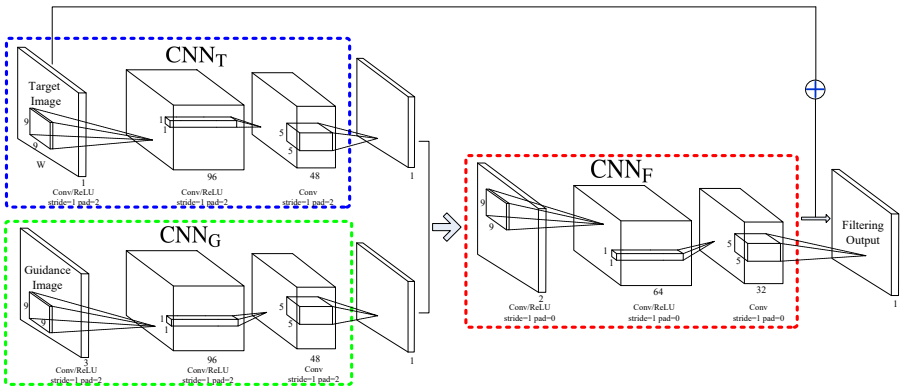


Figure 3.2: Network architecture for joint image filter. The proposed deep joint image filter model consists of three major components. Each component is a three-layer network. In addition, we introduce a skip connection so that the network $\mathrm{CNN_F}$ learns to predict the residuals between the input target image and the desired ground truth output.

ing alleviates the need for restoring specific target image contents (which complicates the learning process), we show significant improvement in transferring accurate details from the guidance to the target image. Second, in [143], we train the model only using an RGB/depth dataset and then evaluate its generalization ability on other domains. In this work, we show that the model trained using an RGB/flow dataset also generalizes well on other visual domains. This demonstrates that our network design is insensitive to the modality of the training data. Third, we evaluate our approach on various joint image filter applications, compare against several state-of-the-art joint image filters (including concurrent work [54, 5]), and conduct a detailed ablation study by analyzing the performance of all methods under different hyper-parameter settings (e.g., filter number, filter size, network depth).

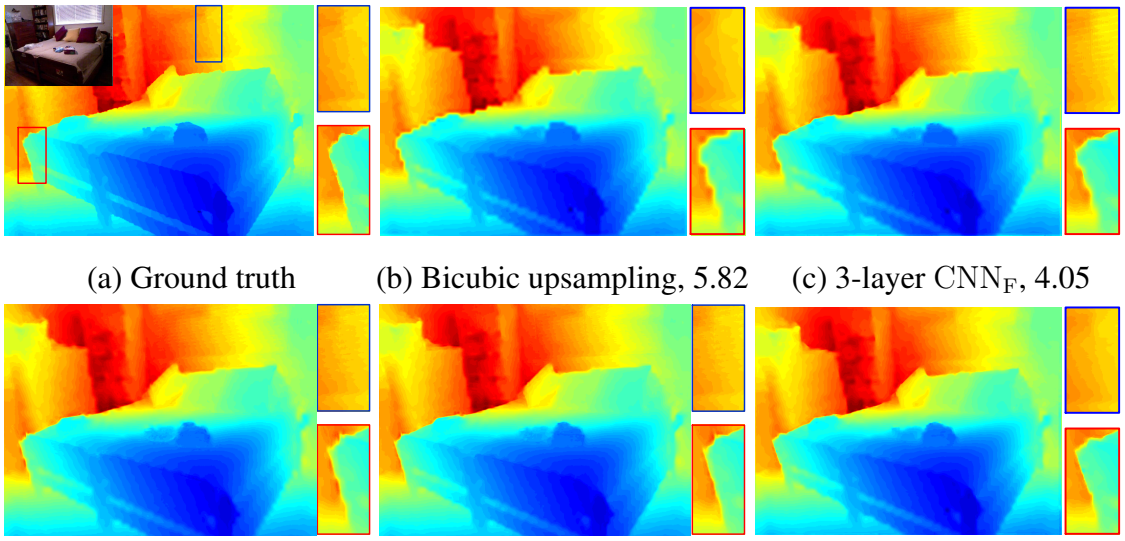
3.2 Learning Joint Image Filters

In this section, we introduce a learning-based joint image filter based on CNNs. We first present the network design (Section 3.2.1) and skip connection (Section 3.2.2). Next, we describe the network training process (Section 3.2.3) and visualize the guidance map generated by the network (Section 3.2.4).

Our CNN model consists of three sub-networks: the target network $\mathrm{CNN_T}$, the guidance network $\mathrm{CNN_G}$, and the filter network $\mathrm{CNN_F}$ as shown in Figure 3.2. First, the sub-network $\mathrm{CNN_T}$ takes the target image as input and extracts a feature map. Second, similar to $\mathrm{CNN_T}$, the sub-network $\mathrm{CNN_G}$ extracts a feature map from the guidance image. Third, the sub-network $\mathrm{CNN_F}$ takes the concatenated feature responses from the sub-networks $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ as input and generates the residual, i.e., the difference between the degraded target image and ground truth. By adding the target input through the skip connection, we obtain the final joint filtering result. Here, the main roles of the two sub-networks $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ are to serve as non-linear feature extractors that capture the local structural details in the respective target and guidance images. The sub-network $\mathrm{CNN_F}$ can be viewed as a non-linear regression function that maps the feature responses from both target and guidance images to the desired residuals. Note that the information from target and guidance images is simultaneously considered when predicting the final filtered result. Such a design allows us to selectively transfer structures and avoid texture-copying artifacts.

3.2.1 Network architecture design

To design a joint filter using CNNs, a straightforward implementation is to concatenate the target and guidance images together and directly train a generic CNN similar to the filter network $\mathrm{CNN_F}$. While in theory, we can train a generic CNN to approximate the desired function for joint filtering, our empirical results show that such a network generates poor results. Figure 3.3(c)-(d) shows an example of joint depth upsampling using



(a) Ground truth (b) Bicubic upsampling, 5.82 (c) 3-layer $\mathrm{CNN_F}$, 4.05

(d) 3-layer $\mathrm{CNN_F\_R}$, 3.96 (e) 4-layer $\mathrm{CNN_F\_R}$, 3.91 (f) Our network, 2.62

Figure 3.3: Comparison of network design. Joint depth upsampling ($8\times$) results of using different network architectures. (a) GT depth map (inset: guidance image). (b) Bicubic upsampling. (c)-(e) Results from the straightforward implementation using $\mathrm{CNN_F}$ and $\mathrm{CNN_F\_R}$. (f) Our results. Note the difference on the bed corner and curtain. The numbers are the RMSE metric based on the GT in (a).

the network $\mathrm{CNN_F}$ and its residual-based variant $\mathrm{CNN_F\_R}$. The main structures (e.g., the bed corner) contained in the guidance image are not well transferred to the target depth image, thereby resulting in blurry boundaries. In addition, inconsistent texture structures in the guidance image (e.g., the stripe pattern of the curtain on the wall) are also incorrectly copied to the target image. A potential approach that may improve the results is to adjust the architecture of $\mathrm{CNN_F}$, such as increasing the network depth or using larger filter sizes. However, as shown in Figure 3.3(e), these variants do not show notable improvement. Blurry boundaries and the texture-copying problem still occur. We note that similar observations have also been reported in [24], which indicate that the effectiveness of deeper structures for low-level tasks is not as apparent as that shown in high-level tasks

(e.g., image classification).

We attribute the limitation of using a generic network for joint filtering to the fact that the original RGB guidance image fails to provide direct and effective guidance as it mixes a variety of information (e.g., texture, intensity, and edges). To validate this intuition, we show in Figure 3.4 one example where we replace the original RGB guidance image with its edge map extracted using [22]. Compared to the results guided by the RGB image (Figure 3.4(d)), the upsampled image using the edge map guidance (Figure 3.4(e)) shows substantial improvement in preserving the sharp edges.

Based on the above observation, we introduce two sub-networks $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ to first construct two separate processing streams for the two images before concatenation. With the proposed architecture, we constrain the network to extract effective features from both images separately first and then fuse them at a later stage to generate the final filtering output. This differs from conventional joint filters where the guidance information is mainly computed from the pixel-level intensity/color differences in the local neighborhoods. As our models are jointly trained in an end-to-end fashion, our result (Figure 3.4(f)) shows further improvements over that of using the edge guided filtering (Figure 3.4(e)).

In this work, we adopt a three-layer structure for each sub-network as shown in Figure 3.2. Given M training image samples $\{I_i^T, I_i^G, I_i^{gt}\}_{i=1}^M$, we learn the network parameters by minimizing the sum of the squared losses:

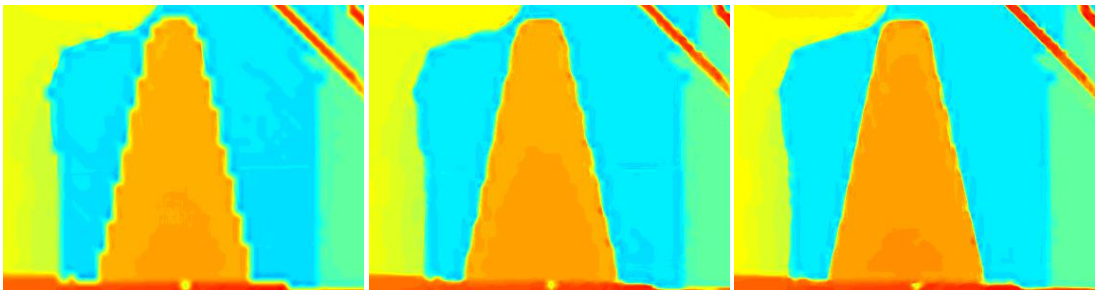
$$\|I^{gt} - \Phi(I^T,\ I^G)\|_2^2\,, \tag{3.1}$$

where Φ denotes the joint image filtering operator. In addition, I^T, I^G, and I^{gt} denote the target image, the guidance image and the ground truth output, respectively.

3.2.2 Skip connection

As the goal of the joint image filter is to leverage the signals from the guidance image to enhance the degraded target image, the input target image and the desired output share the same low-resolution frequency components. We thus introduce a skip connection to

(a) GT depth (b) Guidance (c) Bicubic



(d) RGB guided (e) Edge guided (f) Ours

Figure 3.4: Comparison of different types of guidance. Joint depth upsampling ($8\times$) results using different types of guidance images. Both (d) and (e) are trained using the $\mathrm{CNN_F}$ network. Our method generates sharper boundary of the sculpture (left) and the cone (middle).

enforce the network to focus on learning the residuals rather than predicting the actual pixel values. With the skip connection, the network does not need to learn the identity mapping function from the input target image to the desired output in order to preserve the low-frequency contents. Instead, the network learns to predict the sparse residuals in important regions (e.g., object contours). In Figure 3.5, we show an example of the predicted residuals, which highlights the estimated difference between the target input (Figure 3.5(a)) and the ground truth (Figure 3.5(d)). Quantitative results in Table 3.1 show that with the skip connection, the proposed algorithm obtains notable improvements over [143].

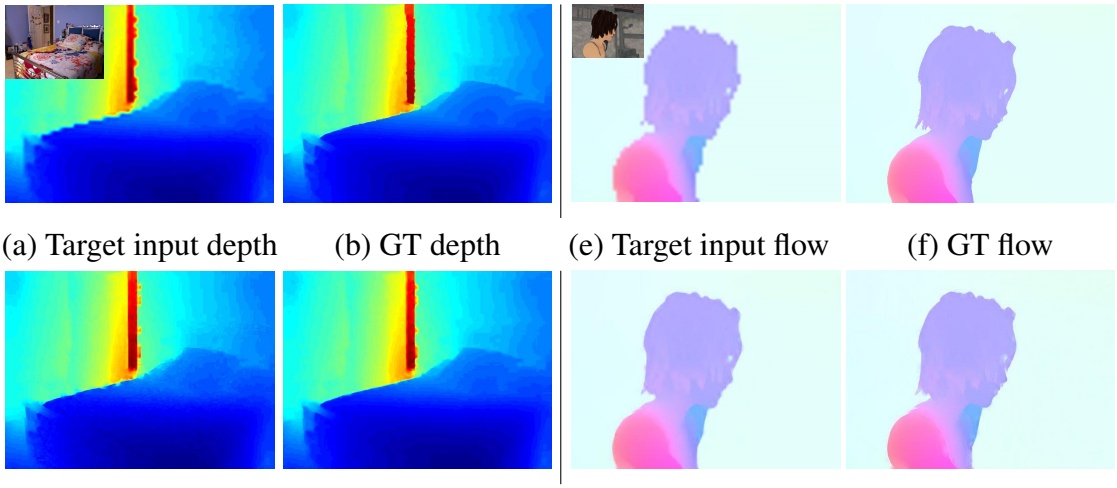
(a) Target input (b) Residual output (c) Filtering output (d) Ground truth

Figure 3.5: Residual prediction. Joint depth upsampling results ($8\times$) of using our network with a skip connection. The filtering output (c) is the summation of (a) the target input and (b) the predicted output.

3.2.3 Network training

Since the target and guidance image pair can be from various modalities (e.g., RGB/depth, RGB/NIR), it is infeasible and costly to collect large datasets and train one network for each type of data pair separately. The goal of our network training, however, is not predicting specific pixel values in one particular modality. Instead, we aim to train the network so that it can selectively transfer structures by leveraging the prior from the guidance image. Hence, we only need to train the network with only one type of image data and then apply the network to other domains.

To demonstrate that the proposed method is insensitive to the training data modality, we train the network with either the RGB/depth dataset [85] or RGB/flow dataset [10]. We conduct a cross-dataset evaluation (training with one type and evaluate on the other) and show the exemplary results in Figure 3.6. Figure 3.6 (a)-(d) shows the upsampled depth maps using models trained with different domains of image data. The flow model refers to the one trained with RGB/flow data for flow map upsampling, while the depth model is trained with RGB/depth data for depth map upsampling. In Figure 3.6(c), we apply the flow model to upsample the degraded depth map and show competitive results obtained by the depth model (Figure 3.6(d)). Similar observations on flow map upsampling are also found in Figure 3.6 (e)-(h). Both the models trained with the flow and depth data achieve



(a) Target input depth (b) GT depth (e) Target input flow (f) GT flow

(c) Upsample by flow model 3.54 (d) Upsample by depth model 3.36 (g) Upsample by depth model 5.63 (h) Upsample by flow model 5.35

Figure 3.6: Effect of training data modalities. (a)-(d) Joint depth map upsampling (8×). The model trained with RGB/flow data generates similar results when compared with the model trained with RGB/depth data. (e)-(h) Joint flow map upsampling (8×). (g) The model trained with RGB/depth data and (h) The model trained with RGB/flow data. The numbers are the RMSE metric comparing against the GT.

similar performance. More filtering results are shown in Section 3.3, where we evaluate the model with different image data from various domains. More quantitative results are presented in Table 3.1.

3.2.4 What has the network learned?

Selective transferring. Using the learned guidance model $\mathrm{CNN_G}$ alone to transfer details may sometimes be erroneous. In particular, the structures extracted from the guidance image may not exist in the target image. The top and middle rows of Figure 3.7 show typical responses at the first layer of $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$. These two sub-networks show

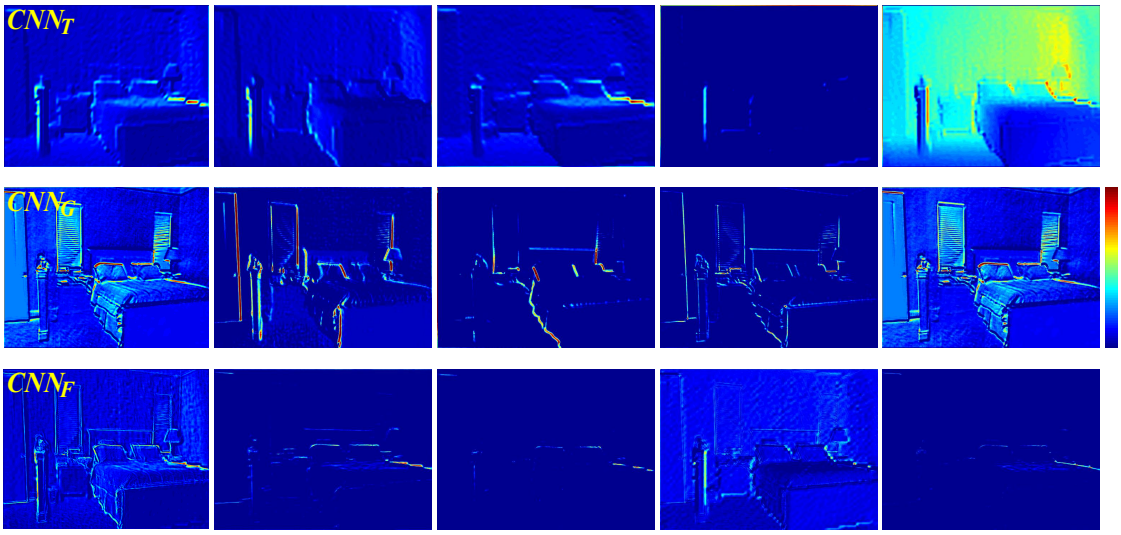
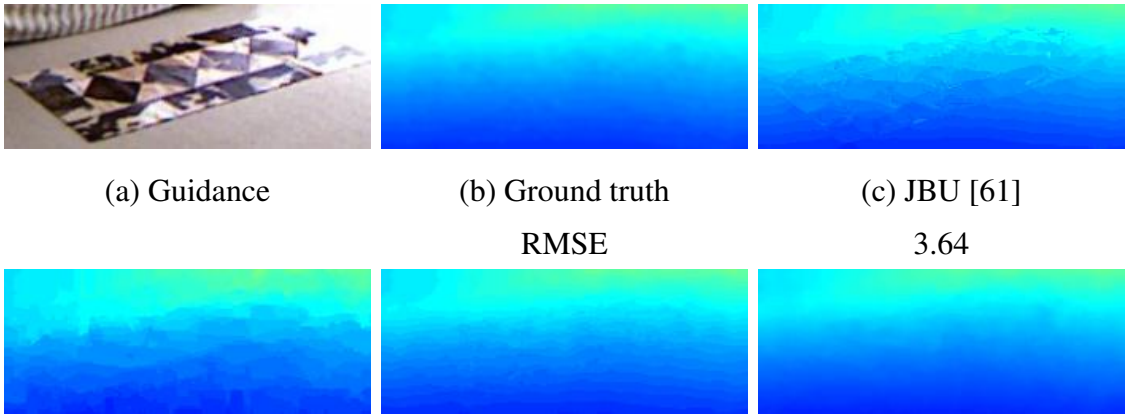


Figure 3.7: Visualization of feature responses. Sample feature responses of the input in Figure 3.9(a) at the first layer of $\mathrm{CNN_T}$ (top) and $\mathrm{CNN_G}$ (middle), and the second layer of $\mathrm{CNN_F}$ (bottom). For each subnetwork, we select five feature channels and visualize the responses through the colormap. The corresponding colorbar is shown in the rightmost. Note that with the help of $\mathrm{CNN_F}$, inconsistent structures (e.g., the window on the wall) are correctly suppressed.

strong responses to edges from the target and guidance images respectively. Note that there are inconsistent structures in the guidance and target images, e.g., the window on the wall. The bottom row of Figure 3.7 shows sample responses at the second layer of $\mathrm{CNN_F}$. We observe that the sub-network $\mathrm{CNN_F}$ suppresses inconsistent details.

We present another example in Figure 3.8. We note that the ground truth depth map of the selected region is smooth. However, due to the high contrast patterns on the mat in the guidance image, several methods, e.g., [61, 88], incorrectly transfer the mat structure to the upsampled depth map. The reason is that these methods [61, 88] rely only on structures in the guidance image. The problem, commonly known as texture-copying artifacts, often occurs when the texture in the guidance image has strong color contrast. With the help



(a) Guidance (b) Ground truth (c) JBU [61]

RMSE 3.64

(d) Park [88] (e) DJF [143] (f) Ours

4.84 3.24 **2.95**

Figure 3.8: Selective transfer. Comparisons of different joint upsampling methods on handling the texture-copying issue. The carpet on the floor contains grid-like texture structures that may be incorrectly transferred to the target image. The numbers are the RMSE metric comparing against the GT.

of the $\mathrm{CNN_F}$, our method successfully blocks the texture structure in the guidance image (Figure 3.8(f)).

Output of $\mathrm{CNN_G}$. In Figure 3.9(c), we show the learned guidance from $\mathrm{CNN_G}$ using two examples from the NYU v2 dataset [85]. In general, the learned guidance appears to be similar to an edge map highlighting the salient structures in the guidance image. We show edge detection results from [22] in Figure 3.9(d). Both results show strong responses to the main structures, but the guidance map generated by $\mathrm{CNN_G}$ appears to detect sharper boundaries while suppressing responses to small-scale textures, e.g., the wall in the first example. The result suggests that using only $\mathrm{CNN_F}$ (Figure 3.3(c)) does not perform well due to lack of the salient feature extraction step from the sub-network $\mathrm{CNN_G}$.

To demonstrate the effectiveness of the skip connection, we compare the learned guidance without and with the skip connection in Figure 3.9(b) and (c). Adding the skip con-

(a) Guidance (inset: GT depth) (b) Learned guidance w/o skip connection (c) Our learned guidance w/ skip connection (d) Edge map [22]

Figure 3.9: Visualization of the learned guidance map. Comparison between the learned guidance feature maps from $\mathrm{CNN_G}$ and edge maps from [22]. The network $\mathrm{CNN_G}$ is capable of extracting informative, salient structures from the guidance image for content transfer. Furthermore, with the skip connection, the learned guidance maps in (c) are cleaner than that in (b) by suppressing inconsistent structures (edges on the window and wall) in the target/guidance pair.

nection helps suppress more inconsistent structures (e.g., edges on the bed, wall, table) in the target/guidance pair, and consequently the residual-based model effectively alleviates texture-copying artifacts.

3.2.5 Relationship to prior work

The proposed framework is closely related to weighted-average, optimization-based, and CNN-based models. In each layer of the network, the convolutional filters also perform the weighted-average process. In this context, our filter is similar to the weighted-average filters. The key difference is that the weights in this work are learned from data while those of the weighted-average filters [110, 61] are pre-defined based on color or gra-

dient features. The proposed network plays a similar role in the fidelity and regularization terms defined in the optimization-based joint filters. Specifically, the training objective in (3.1) corresponds to the fidelity term of the weighted-average filters [110, 61] as it encourages the output to be as close to the ground truth as possible. The skip connection implicitly serves as the regularization term by enforcing adjacent pixels to share similar values (e.g., depth) as it directly bypasses the low-quality target input to the output of the network. For CNN-based models, our network architecture can be viewed as a unified model for different tasks. For example, if we remove $\mathrm{CNN_G}$ and use only $\mathrm{CNN_T}$ and $\mathrm{CNN_F}$, the resulting network architecture resembles an image restoration model, e.g., SRCNN [23]. On the other hand, in cases of removing $\mathrm{CNN_T}$, the remaining $\mathrm{CNN_G}$ and $\mathrm{CNN_F}$ can be viewed as one using CNNs for depth prediction [18].

3.3 Experimental Results

In this section, we demonstrate the effectiveness and applicability of our approach through a broad range of joint image filtering tasks, including joint image upsampling, texture-structure separation, and cross-modality image restoration. The source code and datasets will be made available to the public. More results can be found at `http://vllab1.ucmerced.edu/~yli62/DJF_residual/`.

Network training. To train our network, we randomly collect 160,000 training patch pairs of 32×32 pixels from 1,000 RGB and depth images in the NYU v2 dataset [85]. Images in the NYU dataset are absolute depth maps captured in complicated indoor scenarios. We train two models for two different tasks: (1) joint image upsampling and (2) noise reduction. For the upsampling task, we obtain each low-quality target image from downsampling the ground-truth image (with scale factors of $4\times$, $8\times$, $16\times$) using the nearest neighbor interpolation. For the noise reduction task, we generate the low-quality target image by adding Gaussian noise to each of the ground-truth depth maps with zero mean and variance of 1e-3. We use the MatConvNet toolbox [3] to train our joint filters.

Testing. Using RGB/depth data for training, our model takes a 1-channel target image (depth map) and a 3-channel guidance image (RGB) as inputs. However, the trained model can be applied to other data types in addition to RGB/depth images with simple modifications. For the multi-channel target images, we apply the trained model independently for each channel. For the single-channel guidance images, we replicate it three times to create the 3-channel guidance image.

3.3.1 Depth map upsampling

Datasets. We present quantitative performance evaluation on joint depth upsampling using three benchmark datasets where the corresponding high-resolution RGB images are available:

- Middlebury dataset [98, 49]: We collect 30 images from 2001-2006 datasets with the missing depth values provided by Lu et al. [79].
- NYU v2 dataset [85]: As we use the 1,000 images in this dataset for training, we use the rest of 449 images for testing.
- SUN RGB-D [108]: We use a random subset of 2,000 high-quality RGB/depth image pairs from the 3,784 pairs captured by the Kinect v2 sensor. These images are captured from a variety of complicated indoor scenes.

Note that the data in [85, 108] are *absolute* depth maps representing the physical distances in meters to the observer. However, the data in [98, 49] are *relative* depth maps (disparity), which measure the distance between two corresponding points in a scene under two different views. Each disparity value denotes the number of shifted pixels.

Table 3.1: Quantitative comparisons on depth upsampling. Comparisons with the state-of-the-art methods in terms of RMSE. The depth values are scaled to the range $[0, 255]$ for the Middlebury [98, 49], and SUN RGB-D [108] datasets. For the NYU v2 dataset [85], the depth values are measured in centimeters. Note that the depth maps in the SUN RGB-D dataset may contain missing regions due to the limitation of depth sensors. We ignore these pixels in calculating the RMSE. Numbers in bold indicate the best performance and underscored numbers indicate the second best. The mean of RMSE values are shown in each entry.

	Middlebury [98, 49]			**NYU v2 [85]**			**SUN RGB-D [108]**		
	4×	8×	16×	4×	8×	16×	4×	8×	16×
Bicubic	4.44	7.58	11.87	8.16	14.22	22.32	2.09	3.45	5.48
MRF [20]	4.26	7.43	11.80	7.84	13.98	22.20	1.99	3.38	5.45
GF [46]	4.01	7.22	11.70	7.32	13.62	22.03	1.91	3.31	5.41
JBU [61]	2.44	3.81	6.13	4.07	8.29	13.35	1.37	2.01	3.15
TGV [31]	3.39	5.41	12.03	6.98	11.23	28.13	1.94	3.01	5.87
Park [88]	2.82	4.08	7.26	5.21	9.56	18.10	1.78	2.76	4.77
Ham [44]	3.14	5.03	8.83	5.27	12.31	19.24	1.67	2.60	4.36
DMSG [54]	**1.79**	**3.39**	**5.87**	<u>3.48</u>	<u>6.07</u>	10.27	1.30	<u>1.80</u>	2.81
FBS [5]	2.58	4.19	7.30	4.29	8.94	14.59	1.58	2.27	3.76
Ours-flow	2.31	3.95	6.34	4.42	7.32	11.62	1.36	1.91	2.90
DJF [143]	2.14	3.77	6.12	3.54	6.20	<u>10.21</u>	<u>1.28</u>	1.81	<u>2.78</u>
Ours	<u>1.98</u>	<u>3.61</u>	<u>6.07</u>	**3.38**	**5.86**	**10.11**	**1.27**	**1.77**	**2.75**

Evaluated methods. We compare our model against several state-of-the-art joint image filters for depth map upsampling. The JBU [61], GF [46], Ham [44] and FBS [5] methods are generic joint image upsampling. On the other hand, the MRF [20], TGV [31], Park [88]

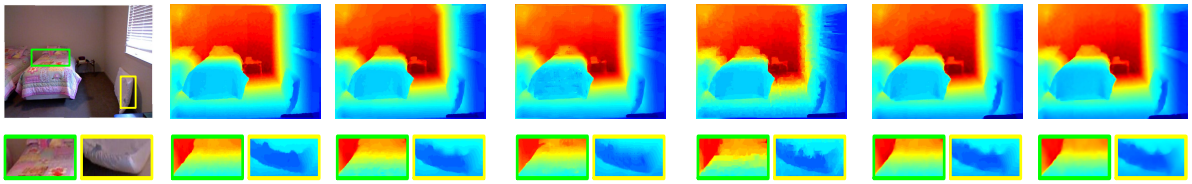
Table 3.2: Run-time performance comparisons. Average run-time of depth map upsampling algorithms on images of size 640×480 pixels.

	MRF	GF	JBU	TGV	Park	Ham	DMSG	FBS	Ours (CPU/GPU)
Time (s)	0.76	0.08	5.64	68.21	45.79	8.62	0.71	0.34	1.31/0.07

and DMSG [54], algorithms are designed specifically for image-guided depth upsampling. Using the experimental protocols for evaluating the joint depth upsampling algorithms [88, 31, 44], we obtain the low-resolution target image from the ground-truth depth map using the nearest-neighbor downsampling method.

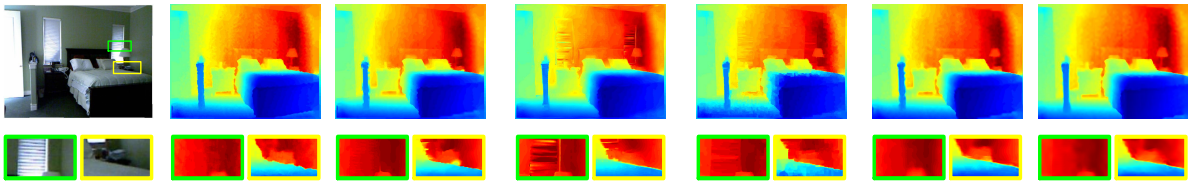
Quantitative comparisons. Table 3.1 shows the quantitative results in terms of the root mean squared errors (RMSE). For other methods, we use the default parameters in the original implementations. The proposed algorithm performs well against the state-of-the-art methods across all three datasets. The extensive evaluations on absolute depth datasets [85, 108] demonstrate the effectiveness of our algorithm in handling complicated real-world indoor scenes. Furthermore, we compare the average run-time of different methods on the NYU v2 dataset in Table 3.2. We carry out all the experiments on the same machine with an Intel i7 3.6GHz CPU and 16GB RAM. We report the running time of our model in either CPU or GPU mode (GTX 745). Among all the evaluated methods, the proposed algorithm is efficient while delivering high-quality upsampling results.

The concurrent DMSG method by Tai et al. [54] outperforms the proposed algorithm on the Middlebury dataset. This can be attributed to several reasons. First, Tai et al. [54] leverage multi-scale guidance data while we use only single scale signals. The multi-scale design requires more network parameters to learn. For example, the model size of the upsampling model ($8\times$) in [54] is 1,822 KB compared to our model size of 526 KB. Second, the model in [54] is trained on a small collection of relative depth maps (82 images) [98, 49].



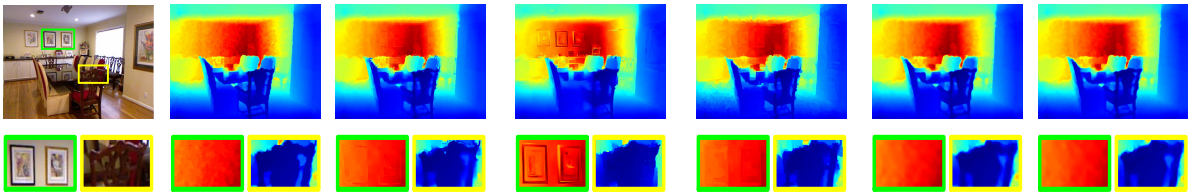
(a) Guidance (b) GT (c) JBU [61] (d) TGV [31] (e) Park [88] (f) DJF[143] (g) Ours

RMSE 4.27 5.62 4.72 3.07 **2.93**



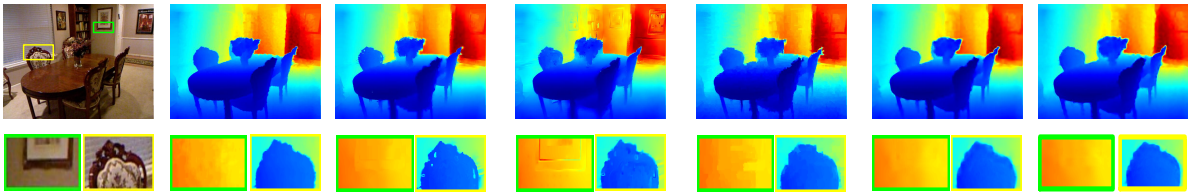
(a) Guidance (b) GT (c) JBU [61] (d) TGV [31] (e) Park [88] (f) DJF[143] (g) Ours

RMSE 4.98 7.24 6.24 3.80 **3.68**



(a) Guidance (b) GT (c) JBU [61] (d) TGV [31] (e) Park [88] (f) DJF[143] (g) Ours

RMSE 11.84 16.05 12.67 7.98 **7.57**



(a) Guidance (b) GT (c) JBU [61] (d) TGV [31] (e) Park [88] (f) DJF[143] (g) Ours

RMSE 9.47 16.23 14.18 7.98 **7.74**

Figure 3.10: Qualitative comparisons on depth upsampling. Comparisons against existing depth upsampling algorithms for a scaling factor of 8×. The numbers (in centimeter) are the RMSE metric comparing against the GT in (b).

In contrast, our model is trained on a large dataset (1000 images) of absolute depth

maps [85]. For fair comparisons using absolute depth maps, we re-train the model of [54] with the same dataset [85] based on our own implementation. Table 3.1 shows that the performance of both [54] and our previous work DJF [143] on absolute depth datasets [85, 108] achieve similar performance. While the method in [54] also uses the similar strategy of predicting residuals, we demonstrate that the proposed algorithm achieves improved results with fewer parameters, suggesting the practical applicability of our model to real-world applications. Another important difference is that the model in [54] is designed only for depth upsampling. Our approach, on the other hand, can be applied to generic joint image filtering tasks.

Effects of skip connection. We validate the contribution of the introduced skip connection by comparing the DJF [143] method and proposed algorithm (bottom two rows of Table 3.1). In Section 3.4, we show that it is difficult to gain further improvement by simply modifying network parameters, such as the filter size, filter number, and network depth. However, with the skip connection, the proposed algorithm obtains significant performance improvement. The performance gain can be explained by that using skip connection alleviates the issues that the network only learns the appearance of the target input images, and helps the network focus on learning the residuals instead.

Effects of training modality. To validate the effect of training with different modalities, we compare our model with a variant that is trained with RGB/flow data (denoted as Ours-flow). We randomly select 1,000 RGB/flow image pairs from the Sintel dataset [10] and collect 80,000 training patch pairs of 32×32 pixels. We use either x-component or y-component of the optical flow as our target image. During the testing phase, we apply the trained model independently for each channel of the target image. Although the model Ours-flow is trained with the RGB/flow data for optical flow upsampling, Ours-flow performs favorably on the task of depth upsampling against our final model (Ours) trained with the RGB/depth data, as shown in Table 3.1.

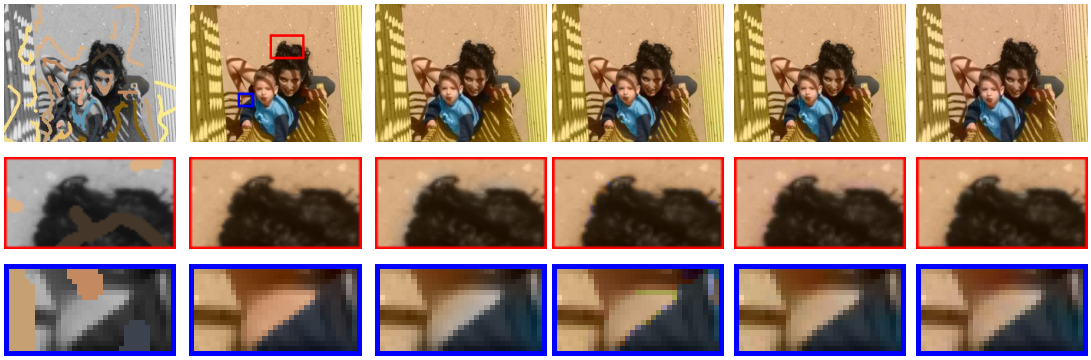
Visual comparisons. We show four examples for qualitative comparisons in Figure 3.10.

It is worth noticing that the proposed joint filter selectively transfers salient structures in the guidance image while avoiding texture-copying artifacts (see the green boxes). The GF [46] method does not recover the degraded boundary well under a large upsampling factor (e.g., $8\times$). The JBU [61], TGV [31] and Park [88] approaches are agnostic to structural consistency between the target and guidance images, and thus transfer erroneous details. In contrast, the results of our algorithm are smoother, sharper and more accurate with respect to the ground truth.

3.3.2 Joint image upsampling

Numerous computational photography applications require obtaining a solution map (e.g., chromaticity, saliency, disparity, labels) over the pixel grid. However, it is often time-consuming or memory-intensive to compute the high-resolution solution maps directly. An alternative is to first obtain a low-res solution map over the downsampled pixel grids and then upsample the low-resolution solution map back to the original resolution with a joint image upsampling algorithm. Such a pipeline requires the upsampling method to restore well image degradation caused by downsampling and avoid the inconsistency issues. In what follows, we demonstrate the use of the learned joint image filters for colorization and saliency as examples. Note that in the following applications we use the *same* model trained with RGB/depth data and evaluate on other image modalities without retraining the network using data in the new domains.

For the colorization task, we first compute the chromaticity map on the downsampled ($4\times$) image using the user-specified color scribbles [67]. We then use the original high-resolution intensity image as the guidance image to jointly upsample the low-resolution chromaticity map. Figure 3.11 shows that our model is able to achieve visually pleasing results with fewer color bleeding artifacts and efficiently. Our results are visually similar to the direct solutions on the high-resolution intensity images (Figure 3.11(b)).



(a) Scribbles	(b) Levin [67]	(c) GF [46]	(d) Ham [44]	(e) DJF [143]	(f) Ours
	RMSE	5.94	6.28	5.57	**5.38**

(a) Scribbles	(b) Levin [67]	(c) GF [46]	(d) Ham [44]	(e) DJF [143]	(f) Ours
	RMSE	5.94	6.28	5.57	**5.38**
Time (s)	8.20	**1.50**	28.8	2.80	2.82

Figure 3.11: Colorization upsampling. Joint image upsampling applied to colorization. We also list the runtime for the colorization upsampling process for each method. The close-up areas show that our joint upsampling results (f) have fewer color bleeding artifacts when compared with other competing algorithms (c-e). Our visual results (f) are comparable with the results computed using the full resolution image in (b). The RMSE metric comparing against the GT in (b) are presented. The average RMSE over all test images are shown in Table 3.3.

(a) Low-res saliency	(b) GF [46]	(c) Ham [44]	(d) DJF [143]	(e) Ours
F-measure	0.722	0.716	0.737	**0.748**

(a) Low-res saliency	(b) GF [46]	(c) Ham [44]	(d) DJF [143]	(e) Ours
F-measure	0.758	0.747	0.772	**0.779**

Figure 3.12: Saliency map upsampling. Visual comparisons of saliency map upsampling results (10×). (a) Low-res saliency map obtained from the downsampled RGB image (inset: guidance image). The numbers are the F-measure metric comparing against the GT. The average F-measure over all test images are shown in Table 3.3.

The quantitative comparisons are presented in the first row of Table 3.3. We use the direct solution of [67] on the high-resolution image as ground truth and compute the RMSE over seven test images in [67]. Table 3.3 shows that our method performs well with the lowest error. Note that our pipeline (low-res result + joint upsampling) is nearly three times faster (2.82 seconds) than directly running the colorization algorithm [67] on the original pixel grid to obtain the high-resolution result (8.20 seconds). Note that for fair comparisons, all run-time results are obtained based on the CPU mode.

For saliency detection, we first compute the saliency map on the downsampled (10×) image using the manifold method by Yang et al. [138]. We then use the original high-resolution intensity image as guidance to upsample the low-resolution saliency map. Figure 3.12 shows the saliency detection results by the state-of-the-art methods and proposed algorithm. Overall, the proposed algorithm generates sharper edges than other alterna-

Table 3.3: Quantitative comparisons of different upsampling methods on difference solution maps.

	Bicubic	GF [46]	Ham [44]	DJF [143]	Ours
RMSE	6.01	5.74	6.31	5.48	5.40
F-measure	0.759	0.766	0.763	0.778	0.781

tives. In addition, we present quantitative evaluation using the ASD benchmark dataset [1] which consists of 1,000 images with manually labeled ground truth. Table 3.3 shows the comparison between different upsampling methods and our approach in terms of F-measure [82]. The experimental results demonstrate that the proposed algorithm performs favorably against the state-of-the-art methods.

3.3.3 Structure-texture separation

We apply our model trained for noise reduction to the task of structure-texture separation. Here we use the target image itself as the guidance. We adopt a similar strategy as in the rolling guidance filter (RGF) [148] to remove small-scale textures, i.e., using the output of the previous iteration as the input of the current iteration.

We use the inverse halftoning task as an example. A halftoned image is generated by the reprographic technique that simulates continuous tone imagery using various dot patterns [62], as shown in Figure 3.13(a). The goal of inverse halftoning is to remove these dots while preserving the main structures. We compare our results with those from the RGF [148], Xu [135], DJF [143] and the method by Kopf [62] for halftoned images reconstruction. Since there exists no ground truth data, we use the results from Kopf [62] as the pseudo ground truth as it is specifically designed for reconstructing halftoned images and achieves the best visual quality. For [148, 135], we carefully select the parameters (listed in Figure 3.13) for the optimal results by considering both removing the dot

(a) Input	(b) Kopf [62]	(c) RGF [148]	(d) Xu [135]	(e) DJF [143]	(f) Ours
	RMSE	7.14	7.30	7.07	**6.91**

(a) Input	(b) Kopf [62]	(c) RGF [148]	(d) Xu [135]	(e) DJF [143]	(f) Ours
	RMSE	16.88	**16.62**	17.86	17.24

Figure 3.13: Inverse halftoning. For each method, we carefully select the parameters for the optimal results. (c) $\sigma_s = 2, \sigma_r = 0.05, iter = 4$. (d) $\lambda = 0.005, \sigma = 1$. (e)-(f) top: $iter = 2$, bottom: $iter = 3$. Since there is no GT, we regard the result of [62] in (b) as the GT because it is an algorithm specifically designed for reconstructing halftoned images. The numbers are the RMSE values computed with respect to (b).

patterns and keeping the sharp edges intact. We use the same high-resolution test images from [62] and present two zoomed-in patch examples in Figure 3.13 for illustration, where one (top) is with small-scale dots and another one (bottom) is with large-scale dots. For the DJF [143] and proposed method, we show the results of running two iterations in the first row and three iterations in the second row of Figure 3.13(e)-(f). Our model achieves better results on removing small-scale dots but worse results on removing large-scale dots compared with the methods in [148, 135]. However, in order to get the best results, both [148, 135] require values computed with respect to optimal parameters for different inputs. Our model (trained on RGB/depth only) is not expected to consistently achieve the best performance but able to generalize well for comparable results on the inverse halftoning task without tuning parameters.

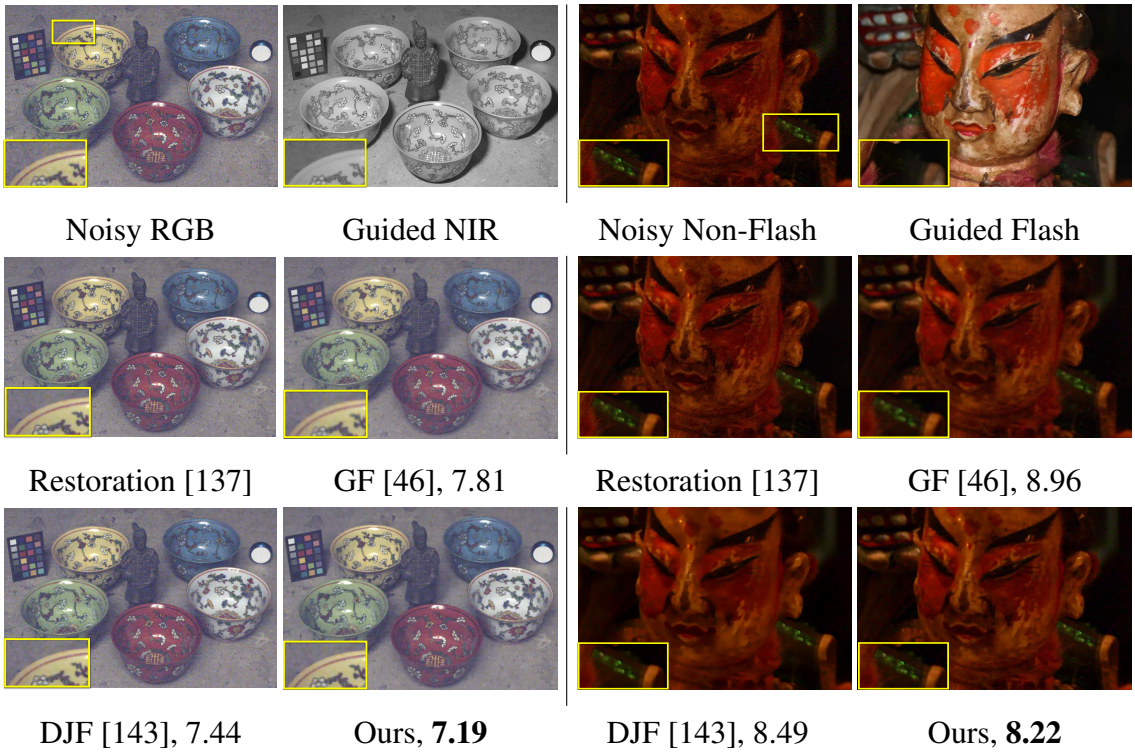


Figure 3.14: Cross-modality filtering for noise reduction. Left: Results of noise reduction using RGB/NIR image pairs (Target: RGB, Guidance: NIR). Right: Results of noise reduction using flash/non-flash image pairs (Target: Non-Flash, Guidance: Flash). The numbers are the RMSE metric comparing against the result of [137].

3.3.4 Cross-modality filtering for noise reduction

Here, we demonstrate that our model can handle various visual domains through two noise reduction applications using RGB/NIR and flash/non-flash image pairs. Figure 3.14 (left) show sample results on joint image denoising with the NIR guidance image. The filtering results by our method are comparable to those of the state-of-the-art technique [137]. For flash/non-flash image pairs, we aim to merge the ambient qualities of the no-flash image with the high-frequency details of the flash image. Guided by a flash image, the filtering result of our method is comparable to that of [137] (Figure 3.14 (right)).

Table 3.4: Quantitative results (RMSE in centimeters for 8×) of using different filter numbers in each sub-network. We apply the same parameters to three sub-networks. Top: without the skip connection, Bottom: with the skip connection.

$n_1 = 256$ $n_2 = 128,$	$n_1 = 128$ $n_2 = 64$	$n_1 = 96$ $n_2 = 48$	$n_1 = 64$ $n_2 = 32$
6.40	6.44	6.32	6.35
5.82	5.84	5.90	5.97

3.4 Discussions

In this section, we first analyze the effects of the performance under different hyper-parameter settings using the network architecture in Figure 3.2. Then, we discuss several limitations of the proposed algorithm. To validate the design choices, we vary the filter number n, filter size f, and depth d of each sub-network. We use the same training process as described in Section 3.3 and evaluate different models on the NYU v2 dataset [85] for 8× upsampling in terms of RMSE.

3.4.1 Filter number

We first analyze the effects of the number of filters (n_1, n_2) in first two layers of each sub-network. The quantitative results are shown in Table 3.4. In the setting of without the skip connection (top row), we observe that larger filter number may not always result in performance improvements because it increases the difficulty of training the network. The results suggest that the performance of such network design is somewhat saturated with the sufficient number of filters. In order to get further improvements, we need to adjust the network design or the learning objectives, rather than simply modifying hyper-parameters.

Such a hypothesis is supported by the setting of with the skip connection, where we add a skip connection to the entire network and reformulate the network as learning residual

functions. The bottom row of Table 3.4 shows that the filter number do yield progressive improvements when it is increased. This is in accordance with the observation in [47, 58] where residual learning is more effective for training the network with larger capacity. However, a larger network also slows down the training process and may only provide marginal performance improvements. Consequently, the selected hyper-parameters of our method (shown in Figure 3.2) strike a good balance between accuracy and computational efficiency.

Furthermore, we discuss the effects of the output channels (n_3) of $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ and show the results in Table 3.5. Intuitively, using multi-dimensional features may improve the model capacity and therefore its performance. However, our experimental results indicate that using multi-dimensional feature maps only slows down the training process without clear performance gain, for both without and with the skip connection settings. Therefore, we set the output feature maps extracted from the target and guidance images as one single channel ($n_3 = 1$).

3.4.2 Filter size

We examine the network sensitivity to the spatial support of the filters. With all the other experimental settings kept the same, we gradually increase the filter size f_i (i=1, 2, 3) in different layers and show the corresponding performance in Table 3.6.

Starting from using small filter sizes ($f_1 = 5$, $f_2 = 1$, $f_3 = 3$), we observe a steady trend of improvements when increasing the filter sizes. This is because smaller filters will restrict the network to focus on detailed local smooth regions that provide little information for restoration. In contrast, a reasonably large filter size can cover richer structural cues that lead to better results. However, when we further enlarge the filter size (e.g.., up to $f_1 = 11$, $f_2 = 3$, $f_3 = 7$), we do not see additional performance gain. We attribute this to the increasing difficulty of network training because larger filter sizes indicate more number of parameters to be learned. Consequently, we choose the filter size $f_1 = 9$,

Table 3.5: Quantitative results (RMSE in centimeters for 8×) of using different filter numbers in the 3rd layer of $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$. Top: without the skip connection, Bottom: with the skip connection.

$n_3 = 1$	$n_3 = 16$	$n_3 = 32$	$n_3 = 64$
6.20	6.40	6.24	6.34
5.86	6.11	5.93	6.02

Table 3.6: Quantitative results (RMSE in centimeters for 8×) of using different filter sizes in each sub-network. Top: without the skip connection, Bottom: w/ the skip connection.

$f_1 = 11$ $f_2 = 3$ $f_3 = 7$	$f_1 = 9$ $f_2 = 3$ $f_3 = 7$	$f_1 = 9$ $f_2 = 1$ $f_3 = 5$	$f_1 = 7$ $f_2 = 1$ $f_3 = 5$	$f_1 = 5$ $f_2 = 1$ $f_3 = 3$
6.28	6.40	6.20	6.47	6.62
5.93	6.05	5.86	6.06	6.24

$f_2 = 1$, and $f_3 = 5$ as a good trade-off between the efficiency and performance.

3.4.3 Network depth

As suggested in [24] that the number of layers does not play a significant role in non-residual based models for low-level tasks, we focus on evaluating the residual-based model (with the skip connection) with different network depth. First, we analyze whether using one generic but deeper residual-based $\mathrm{CNN_F\_R}$ network can improve the performance. We gradually increase the depth from 3 to 8 and show the results in Table 3.7. Overall, the performance of the $\mathrm{CNN_F\_R}$ network improves with a deeper network. However, the performance quickly reaches the point of diminishing returns after d is larger than 4.

Table 3.7: Quantitative evaluation (RMSE in centimeters for $8\times$) when using residual-based $\mathrm{CNN_F\_R}$ only under different network depth d.

$d=3$	$d=4$	$d=5$	$d=6$	$d=7$	$d=8$	Ours
6.31	6.25	6.22	6.20	6.17	6.16	5.86

Table 3.8: Quantitative evaluation of our model by increasing the number of layers (the depth d) used in each subnetwork.

	$d=2$	$d=3$	$d=4$	$d=5$
RMSE / cm	5.99	5.86	5.77	5.73
Model size / MB	0.48	0.53	5.0	11.4

Next, we evaluate our model (three subnetworks) by increasing the network depth. We simultaneously increase the depth d of each subnetwork from 2 to 5 and show the corresponding results in Table 3.8. We observe that equipped with the skip connection a deeper network generally leads to better performance. This is in accordance with the observation in [58] where a 20-layer deep residual net is used for image super-resolution. However, in our case with three subnetworks, the deeper network also induces fast growth of model size as well as longer training time. We find the performance improvement is incremental when d is varied from 3 to 5. Thus, we set d to 3 as a trade-off between model size and performance.

3.4.4 Merging layer

As shown in Figure 3.2, the $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ are merged at the output (third) layer. Here we further analyze the effect of merging $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ at different layers. We fix the whole network depth as 6 and analyze different combinations of network depth

Table 3.9: Quantitative evaluation of different combinations of network depth of $\mathrm{CNN_T}$ ($\mathrm{CNN_G}$) and $\mathrm{CNN_F}$.

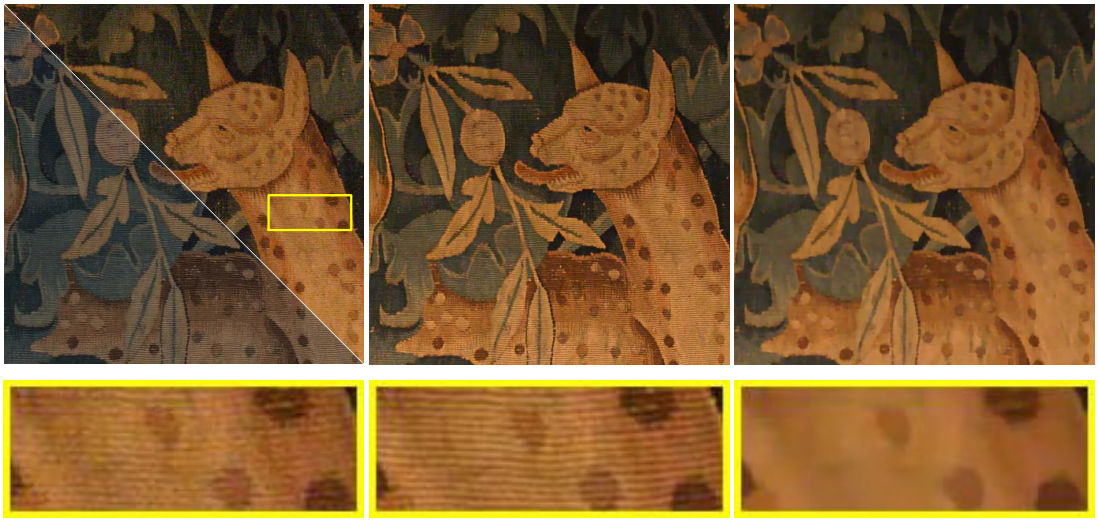
$\mathrm{CNN_T}/\mathrm{CNN_G} - \mathrm{CNN_F}$	0/0 – 6	2/2 – 4	3/3 – 3	4/4 – 2
RMSE / cm	6.13	5.95	5.86	6.03

of $\mathrm{CNN_T}$, $\mathrm{CNN_G}$ and $\mathrm{CNN_F}$. We gradually increase the depth of $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ while decreasing the depth of $\mathrm{CNN_F}$ (in order to maintain the overall network depth). For example, $0/0 - 6$ (Table 3.9) indicates that we directly stack the target and guidance image and applying a 6-layer $\mathrm{CNN_F}$ only. The evaluation results of different models are shown in Table 3.9. Overall, deeper target/guidance networks ($\mathrm{CNN_T}$ and $\mathrm{CNN_G}$) result in sizable performance improvement resulting from effective feature extractions. However, as the $\mathrm{CNN_F}$ becomes shallower, the performance degrades again. This indicates that neither the $\mathrm{CNN_T}$ ($\mathrm{CNN_G}$) nor the $\mathrm{CNN_F}$ should be too shallow. Therefore, we chose the combination of $3/3 - 3$ for best performance.

3.4.5 Limitations

We note that in some images, our model fails to transfer small-scale details from the guidance map. In such cases, our model incorrectly treats certain small-scale details as noise. This can be explained by the fact that our training data is based on depth images that are mostly smooth and does not contain many spatial details.

Figure 3.15 shows two examples of a flash/non-flash pair for noise reduction. There are several spotty textures on the porcelain in the guided flash image that should have been preserved when filtering the noisy non-flash image. Similarly, our method is not able to effectively transfer the small-scale strip textures on the carpet to the target image. Compared with the method by Georg et al. [38] (Figure 3.15(b) and (d)) that is designed specifically for flash/non-flash images, our filter treats these small-scale details as noise



(a) Input (b) Georg et al. [38] (c) Ours

Figure 3.15: Failure cases. Detailed small-scale textures (yellow rectangle) in the guidance image are over-smoothed by our filter.

and tends to over-smooth the contents. We will collect more training data from other domains (e.g., flash/non-flash) to address the over-smoothing problem in our future work.

3.5 Conclusions

In this work, we present a learning-based approach for joint filtering based on convolutional neural networks. Instead of relying only on the guidance image, we design two sub-networks $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ to extract informative features from both the target and guidance images. These feature maps are then concatenated as inputs for the network $\mathrm{CNN_F}$ to selectively transfer salient structures from the guidance image to the target image while suppressing structures that are not consistent in both images. While we train our network on one type of data (RGB/depth or RGB/flow), our model generalizes well on handling images in various modalities, e.g., RGB/NIR and flash/non-Flash image pairs.

We show that the proposed algorithm is computationally efficient and performs favorably against the state-of-the-art techniques on a wide variety of computer vision and computational photography applications, including cross-modal denoising, joint image upsampling, and texture-structure separation.

Chapter 4

Universal Style Transfer

Universal style transfer aims to transfer arbitrary visual styles to content images. Existing feed-forward based methods, while enjoying the inference efficiency, are mainly limited by inability of generalizing to unseen styles or compromised visual quality. In this work, we present a simple yet effective method that tackles these limitations without training on any pre-defined styles. The key ingredient of our method is a pair of feature transforms, whitening and coloring, that are embedded to an image reconstruction network. The whitening and coloring transforms reflect a direct matching of feature covariance of the content image to a given style image, which shares similar spirits with the optimization of Gram matrix based cost in neural style transfer. We demonstrate the effectiveness of our algorithm by generating high-quality stylized images with comparisons to a number of recent methods. We also analyze our method by visualizing the whitened features and synthesizing textures via simple feature coloring.

4.1 Introduction

Style transfer is an important image editing task which enables the creation of new artistic works. Given a pair of examples, i.e., the content and style image, it aims to synthesize an image that preserves some notion of the content but carries characteristics of the style. The key challenge is how to extract effective representations of the style and then match it in the content image. The seminal work by Gatys et al. [35, 36] show that the correlation between features, i.e., Gram matrix or covariance matrix (shown to be as effective as Gram matrix in [141]), extracted by a trained deep neural network has remarkable ability of capturing visual styles. Since then, significant efforts have been made to synthesize stylized images by minimizing Gram/covariance matrices based loss functions, through either iterative optimization [36] or trained feed-forward networks [114, 56, 141, 13, 27]. Despite the recent rapid progress, these existing works often trade off between generalization, quality and efficiency, which means that optimization-based methods can handle arbitrary styles with pleasing visual quality but at the expense of high computational costs, while feed-forward approaches can be executed efficiently but are limited to a fixed number of styles or compromised visual quality.

By far, the problem of universal style transfer remains a daunting task as it is challenging to develop neural networks that achieve generalization, quality and efficiency at the same time. The main issue is how to properly and effectively apply the extracted style characteristics (feature correlations) to content images in a style-agnostic manner.

In this work, we propose a simple yet effective method for universal style transfer, which enjoys the style-agnostic generalization ability with marginally compromised visual quality and execution efficiency. The transfer task is formulated as image reconstruction processes, with the content features being *transformed* at intermediate layers with regard to the statistics of the style features, in the midst of feed-forward passes. In each intermediate layer, our main goal is to transform the extracted content features such that they exhibit the same statistical characteristics as the style features of the same layer and we found that

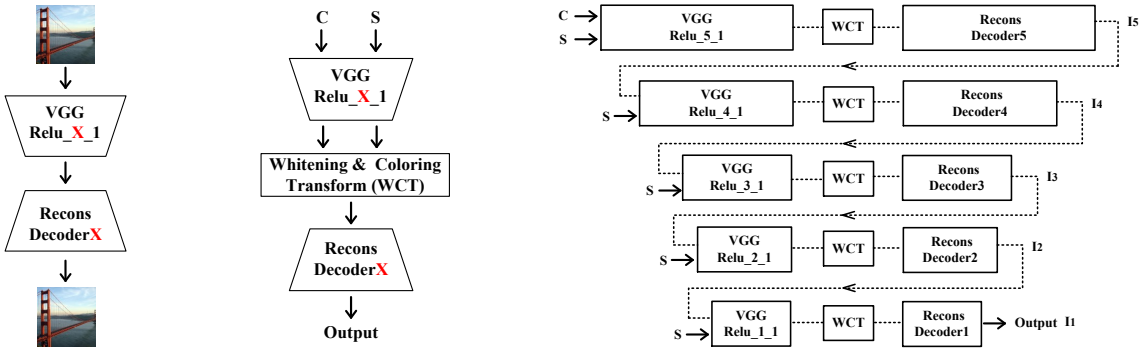
the classic signal *whitening* and *coloring* transforms (WCTs) on those features are able to achieve this goal in an almost effortless manner.

In this work, we first employ the VGG-19 network [105] as the feature extractor (encoder), and train a symmetric decoder to invert the VGG-19 features to the original image, which is essentially the image reconstruction task (Figure 4.1(a)). Once trained, both the encoder and the decoder are fixed through all the experiments. To perform style transfer, we apply WCT to one layer of content features such that its covariance matrix matches that of style features, as shown in Figure 4.1(b). The transformed features are then fed forward into the downstream decoder layers to obtain the stylized image. In addition to this single-level stylization, we further develop a multi-level stylization pipeline, as depicted in Figure 4.1(c), where we apply WCT sequentially to multiple feature layers. The multi-level algorithm generates stylized images with greater visual quality, which are comparable or even better with much less computational costs. We also introduce a control parameter that defines the degree of style transfer so that the users can choose the balance between stylization and content preservation. The entire procedure of our algorithm only requires learning the image reconstruction decoder with *no* style images involved. So when given a new style, we simply need to extract its feature covariance matrices and apply them to the content features via WCT. Note that this learning-free scheme is fundamentally different from existing feed-forward networks that require learning with pre-defined styles and fine-tuning for new styles. Therefore, our approach is able to achieve style transfer universally.

The main contributions of this work are summarized as follows:

- We propose to use feature transforms, i.e., whitening and coloring, to directly match content feature statistics to those of a style image in the deep feature space.

- We couple the feature transforms with a pre-trained general encoder-decoder network, such that the transferring process can be implemented by simple feed-forward operations.



(a) Reconstruction (b) Single-level stylization (c) Multi-level stylization

Figure 4.1: Universal style transfer pipeline. (a) We first pre-train five decoder networks DecoderX (X=1,2,...,5) through image reconstruction to invert different levels of VGG features. (b) With both VGG and DecoderX *fixed*, and given the content image C and style image S, our method performs the style transfer through whitening and coloring transforms. (c) We extend single-level to multi-level stylization in order to match the statistics of the style at all levels. The result obtained by matching higher level statistics of the style is treated as the new content to continue to match lower-level information of the style.

- We demonstrate the effectiveness of our method for universal style transfer with high-quality visual results, and also show its application to universal texture synthesis.

4.2 Proposed Algorithm

We formulate style transfer as an image reconstruction process coupled with feature transformation, i.e., whitening and coloring. The reconstruction part is responsible for inverting features back to the RGB space and the feature transformation matches the statistics of a content image to a style image.

4.2.1 Reconstruction decoder

We construct an auto-encoder network for general image reconstruction. We employ the VGG-19 [105] as the encoder, fix it and train a decoder network simply for inverting VGG features to the original image, as shown in Figure 4.1(a). The decoder is designed as being symmetrical to that of VGG-19 network (up to Relu_X_1 layer), with the nearest neighbor upsampling layer used for enlarging feature maps. To evaluate with features extracted at different layers, we select feature maps at five layers of the VGG-19, i.e., Relu_X_1 (X=1,2,3,4,5), and train five decoders accordingly. The pixel reconstruction loss [25] and feature loss [56, 25] are employed for reconstructing an input image,

$$L = \|I_o - I_i\|_2^2 + \lambda \|\Phi(I_o) - \Phi(I_i)\|_2^2 \,, \tag{4.1}$$

where I_i, I_o are the input image and reconstruction output, and Φ is the VGG encoder that extracts the Relu_X_1 features. In addition, λ is the weight to balance the two losses. After training, the decoder is fixed (i.e., will not be fine-tuned) and used as a feature inverter.

4.2.2 Whitening and coloring transforms

Given a pair of content image I_c and style image I_s, we first extract their vectorized VGG feature maps $f_c \in \Re^{C \times H_c W_c}$ and $f_s \in \Re^{C \times H_s W_s}$ at a certain layer (e.g., Relu_4_1), where H_c, W_c (H_s, W_s) are the height and width of the content (style) feature, and C is the number of channels. The decoder will reconstruct the original image I_c if f_c is directly fed into it. We next propose to use a whitening and coloring transform to adjust f_c with respect to the statistics of f_s. The goal of WCT is to directly transform the f_c to match the covariance matrix of f_s. It consists of two steps, i.e., whitening and coloring transform.

Whitening transform. Before whitening, we first center f_c by subtracting its mean vector m_c. Then we transform f_c linearly as in (4.2) so that we obtain $\hat{f}_c$ such that the feature

Figure 4.2: Inverting whitened features. We invert the whitened VGG Relu_4_1 feature as an example. Left: original images, Right: inverted results (pixel intensities are rescaled for better visualization). The whitened features still maintain global content structures.

maps are uncorrelated ($\hat{f}_c \hat{f}_c^{\top} = I$),

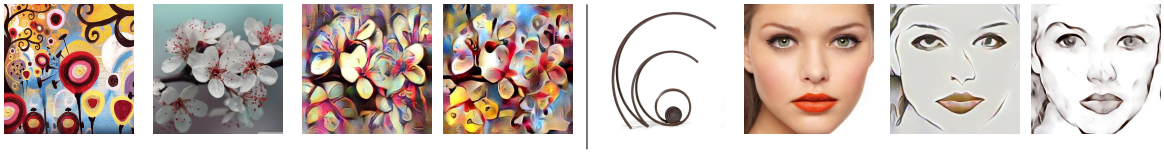
$$\hat{f}_c = E_c \, D_c^{-\frac{1}{2}} \, E_c^{\top} \, f_c \, , \tag{4.2}$$

where D_c is a diagonal matrix with the eigenvalues of the covariance matrix $f_c \, f_c^{\top} \in \Re^{C \times C}$, and E_c is the corresponding orthogonal matrix of eigenvectors, satisfying $f_c \, f_c^{\top} = E_c D_c E_c^{\top}$.

To validate what is encoded in the whitened feature $\hat{f}_c$, we invert it to the RGB space with our previous decoder trained for reconstruction only. Figure 4.2 shows two visualization examples, which indicate that the whitened features still maintain global structures of the image contents, but greatly help remove other information related to styles. We note especially that, for the *Starry_night* example on right, the detailed stroke patterns across the original image are gone. In other words, the whitening step helps peel off the style from an input image while preserving the global content structure. The outcome of this operation is ready to be transformed with the target style.

Coloring transform. We first center f_s by subtracting its mean vector m_s, and then carry out the coloring transform [51], which is essentially the inverse of the whitening step to transform $\hat{f}_c$ linearly as in (4.3) such that we obtain $\hat{f_{cs}}$ which has the desired correlations between its feature maps ($\hat{f_{cs}} \, \hat{f_{cs}}^{\top} = f_s \, f_s^{\top}$),

$$\hat{f_{cs}} = E_s \, D_s^{\frac{1}{2}} \, E_s^{\top} \, \hat{f}_c \, , \tag{4.3}$$



(a) Style (b) Content (c) HM (d) WCT (e) Style (f) Content (g) HM (h) WCT

Figure 4.3: Comparisons between different feature transform strategies. Results are obtained by our multi-level stylization framework in order to match all levels of information of the style.

where D_s is a diagonal matrix with the eigenvalues of the covariance matrix $f_s\ f_s^\top \in \Re^{C\times C}$, and E_s is the corresponding orthogonal matrix of eigenvectors. Finally we re-center the $\hat{f_{cs}}$ with the mean vector m_s of the style, i.e., $\hat{f_{cs}} = \hat{f_{cs}} + m_s$.

To demonstrate the effectiveness of WCT, we compare it with a commonly used feature adjustment technique, i.e., histogram matching (HM), in Figure 4.3. The channel-wise histogram matching [41] method determines a mapping function such that the mapped f_c has the same cumulative histogram as f_s. In Figure 4.3, it is clear that the HM method helps transfer the global color of the style image well but fails to capture salient visual patterns, e.g., patterns are broken into pieces and local structures are misrepresented. In contrast, our WCT captures patterns that reflect the style image better. This can be explained by that the HM method does not consider the correlations between features channels, which are exactly what the covariance matrix is designed for.

After the WCT, we may blend $\hat{f_{cs}}$ with the content feature f_c as in (4.4) before feeding it to the decoder in order to provide user controls on the strength of stylization effects:

$$\hat{f_{cs}} = \alpha\ \hat{f_{cs}} + (1-\alpha)\ f_c\ , \tag{4.4}$$

where α serves as the style weight for users to control the transfer effect.

(a) Style (b) Relu_1_1 (c) Relu_2_1 (d) Relu_3_1 (e) Relu_4_1 (f) Relu_5_1

Figure 4.4: Single-level stylization using different VGG features. The content image is from Figure 4.2.

(a) I_5 (b) I_4 (c) I_1 (d) Fine-to-coarse

Figure 4.5: (a)-(c) Intermediate results of our coarse-to-fine multi-level stylization framework in Figure 4.1(c). The style and content images are from Figure 4.4. I_1 is the final output of our multi-level pipeline. (d) Reversed fine-to-coarse multi-level pipeline.

4.2.3 Multi-level coarse-to-fine stylization

Based on the single-level stylization framework shown in Figure 4.1(b), we use different layers of VGG features Relu_X_1 (X=1,2,...,5) and show the corresponding stylized results in Figure 4.4. It clearly shows that the higher layer features capture more complicated local structures, while lower layer features carry more low-level information (e.g., colors). This can be explained by the increasing size of receptive field and feature complexity in the network hierarchy. Therefore, it is advantageous to use features at all five layers to fully capture the characteristics of a style from low to high levels.

Figure 4.1(c) shows our multi-level stylization pipeline. We start by applying the WCT

Table 4.1: Differences between our approach and other methods.

	Chen et al. [14]	Huang et al. [53]	TNet [114]	DeepArt [36]	Ours
Arbitrary	√	√	×	√	√
Efficient	√	√	√	×	√
Learning-free	×	×	×	√	√

on Relu_5_1 features to obtain a coarse stylized result and regard it as the new content image to further adjust features in lower layers. An example of intermediate results are shown in Figure 4.5. We show the intermediate results I_5, I_4, I_1 with obvious differences, which indicates that the higher layer features first capture salient patterns of the style and lower layer features further improve details. If we reverse feature processing order (i.e., fine-to-coarse layers) by starting with Relu_1_1, low-level information cannot be preserved after manipulating higher level features, as shown in Figure 4.5(d).

4.3 Experimental Results

4.3.1 Decoder training

For the multi-level stylization approach, we separately train five reconstruction decoders for features at the VGG-19 Relu_X_1 (X=1,2,...,5) layer. It is trained on the Microsoft COCO dataset [73] and the weight λ to balance two losses in (4.1) is set as 1.

4.3.2 Style transfer

To demonstrate the effectiveness of the proposed algorithm, we list the differences with existing methods in Table 4.1 and present stylized results in Figure 4.6.

(a) Style (b) [14] (c) [53] (d) [114] (e) [36] (f) Ours

Figure 4.6: Results from different style transfer methods. The content images are from Figure 4.2-4.3. We evaluate various styles including paintings, abstract styles, and styles with obvious patterns.

The optimization-based work of [36] handles arbitrary styles but is likely to encounter

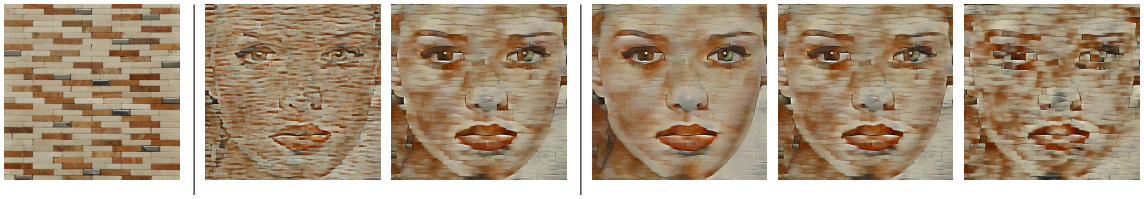


Figure 4.7: Controlling the stylization on the scale and weight.

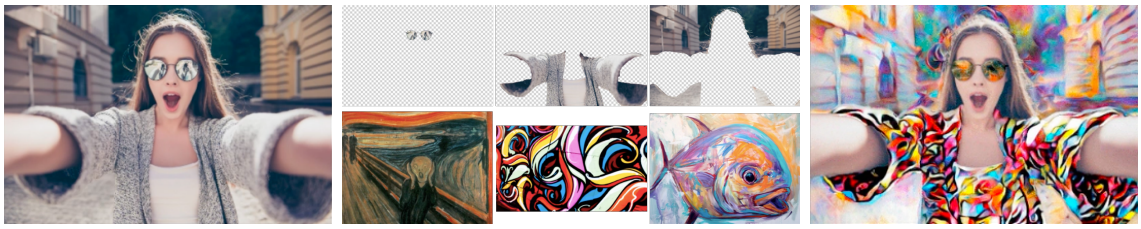
unexpected local minima issues (e.g., 5th and 6th row of Figure 4.6(e)). Although the method [114] greatly improves the stylization speed, it trades off quality and generality for efficiency, which generates repetitive patterns that overlay with the image contents (Figure 4.6(d)). Closest to our work on generalization are the recent methods [14, 53], but the quality of the stylized results are less appealing. The work of [14] replaces the content feature with the most similar style feature based on patch similarity and hence has limited capability, i.e., the content is strictly preserved while style is not well reflected with only low-level information (e.g., colors) transferred, as shown in Figure 4.6(b). In [53], the content feature is simply adjusted to have the same mean and variance with the style feature, which is not effective in capturing high-level representations of the style. Even learned with a set of training styles, it does not generalize well on unseen styles. Results in Figure 4.6(c) indicate that the method in [53] is not effective at capturing and synthesizing salient style patterns, especially for complicated styles where there are rich local structures and non-smooth regions.

Figure 4.6(f) shows the stylized results of our approach. Without learning any style, our method is able to capture visually salient patterns in style images (e.g., the brick wall on the 6th row). Moreover, key components in the content images (e.g., bridge, eye, mouth) are also well stylized in our results, while other methods only transfer patterns to relatively smooth regions (e.g., sky, face). The models and code are available at `https://github.com/Yijunmaverick/UniversalStyleTransfer`.

In addition, we quantitatively evaluate different methods by computing the covariance

Table 4.2: Quantitative comparisons between different stylization methods in terms of the covariance matrix difference (L_s), user preference and run-time, tested on images of size 256×256 and a 12GB TITAN X.

	Chen et al. [14]	Huang et al. [53]	TNet [114]	Gatys et al. [36]	Ours
log(L_s)	7.4	7.0	6.8	6.7	**6.3**
Preference/%	15.7	24.9	12.7	16.4	**30.3**
Time/sec	2.1	0.20	0.18	21.2	**0.83**



(a) Content (b) Different masks and styles (c) Our result

Figure 4.8: Spatial control in transferring, which enables users to edit the content with different styles.

matrix difference (L_s) on all five levels of VGG features between stylized results and the given style image. We randomly select 10 content images from [73] and 40 style images from [57], compute the averaged difference over all styles, and show the results in Table 4.5 (1st row). Quantitative results show that we generate stylized results with lower L_s, i.e., closer to the statistics of the style.

User study. Evaluating artistic style transfer has been an open question in the community. Since the qualitative assessment is highly subjective, we conduct a user study to evaluate 5 methods shown in Figure 4.6. We use 5 content images and 30 style images, and generate 150 results based on each content/style pair for each method. We randomly select 15 style images for each subject to evaluate. We display stylized images by 5 com-

pared methods side-by-side on a webpage in random order. Each subject is asked to vote his/her ONE favorite result for each style. We finally collect the feedback from 80 subjects of totally 1,200 votes and show the percentage of the votes each method received in Table 4.5 (2nd row). The study shows that our method receives the most votes for better stylized results. It can be an interesting direction to develop evaluation metrics based on human visual perception for general image synthesis problems.

Efficiency. In Table 4.5 (3rd row), we also compare our approach with other methods in terms of efficiency. The method by Gatys et al. [36] is slow due to loops of optimization and usually requires at least 500 iterations to generate good results. The methods [114] and [53] are efficient as the scheme is based on one feed-forward pass with a trained network. The approach [14] is feed-forward based but relatively slower as the feature swapping operation needs to be carried out for thousands of patches. Our approach is also efficient but a little bit slower than [114, 53] because we have a eigenvalue decomposition step in WCT. But note that the computational cost on this step will not increase along with the image size because the the dimension of covariance matrix only depends on filter numbers (or channels), which is at most 512 (Relu_5_1). Currently the decomposition step is implemented based on CPU. Our future work includes more efficient GPU implementations of the proposed algorithm.

User Controls. Given a content/style pair, our approach is not only as simple as a one-click transferring, but also flexible enough to accommodate different requirements from users by providing different controls on the stylization, including the scale, weight and spatial control. The style input on different scales will lead to different extracted statistics due to the fixed receptive field of the network. Therefore the scale control is easily achieved by adjusting the style image size. In the middle of Figure 4.7, we show two examples where the brick can be transferred in either small or large scale. The weight control refers to controlling the balance between stylization and content preservation. As shown

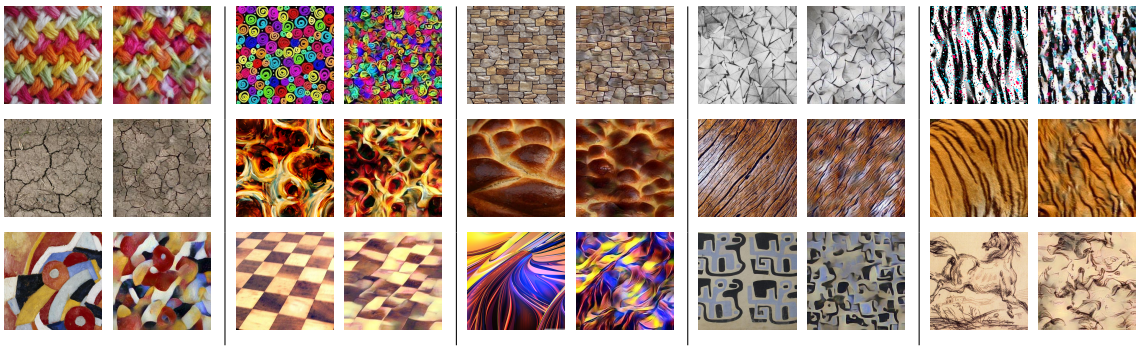


Figure 4.9: Texture synthesis. In each panel, Left: original textures, Right: our synthesized results. Texture images are mostly from the Describable Textures Dataset (DTD) [16].

on right of Figure 4.7, our method enjoys this flexibility in simple feed-forward passes by simply adjusting the style weight α in (4.4). However in [36] and [114], to obtain visual results of different weight settings, a new round of time-consuming optimization or model training is needed. Moreover, our blending directly works on deep feature space before inversion/reconstruction, which is fundamentally different from [36, 114] where the blending is formulated as the weighted sum of the content and style losses that may not always lead to a good balance point.

The spatial control is also highly desired when users want to edit an image with different styles transferred on different parts of the image. Figure 4.8 shows an example of spatially controlling the stylization. A set of masks M (Figure 4.8(b)) is additionally required as input to indicate the spatial correspondence between content regions and styles. By replacing the content feature f_c in (4.3) with $M \odot f_c$ where $\odot$ is a simple mask-out operation, we are able to stylize the specified region only.

4.3.3 Texture synthesis

By setting the content image as a random noise image (e.g., Gaussian noise), our stylization framework can be easily applied to texture synthesis. An alternative is to directly initialize the $\hat{f}_c$ in (4.3) to be white noise. Both approaches achieve similar results. Fig-

Figure 4.10: Interpolation between two texture examples. Left: original textures, Middle: our interpolation results, Right: interpolated results of [36]. β controls the weight of interpolation.

ure 4.9 shows a few examples of the synthesized textures. We empirically find that it is better to run the multi-level pipeline for a few times (e.g., 3) to get more visually pleasing results.

Our method is also able to synthesize the interpolated result of two textures. Given two texture examples s_1 and s_2, we first perform the WCT on the input noise and get transformed features $\hat{f_{cs_1}}$ and $\hat{f_{cs_2}}$ respectively. Then we blend these two features $\hat{f_{cs}} = \beta\ \hat{f_{cs_1}} + (1\text{-}\beta)\ \hat{f_{cs_2}}$ and feed the combined feature into the decoder to generate mixed effects. Note that our interpolation directly works on deep feature space. By contrast, the method in [36] generates the interpolation by matching the weighted sum of Gram matrices of two textures at the loss end. Figure 4.10 shows that the result by [36] is simply overlaid by two textures while our method generates new textural effects, e.g., bricks in the stripe shape.

One important aspect in texture synthesis is *diversity*. By sampling different noise images, our method can generate diverse synthesized results for each texture. While [114] can generate different results driven by the input noise, the learned networks are very likely to be trapped in local optima. In other words, the noise is marginalized out and thus fails to drive the network to generate large visual variations. In contrast, our approach explains each input noise better because the network is unlikely to absorb the variations in input noise since it is never trained for learning textures. We compare the diverse outputs of our model with [114] in Figure 4.11. Note that the common diagonal layout is shared across

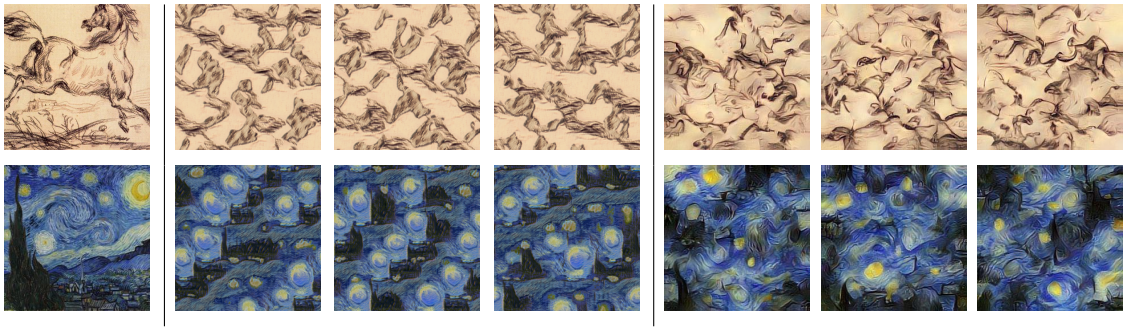


Figure 4.11: Comparisons of diverse synthesized results between TNet [114] and our model.

different results of [114], which causes unsatisfying visual experiences. The comparison shows that our method achieves diversity in a more natural and flexible manner.

4.4 Extension to Photo Style

Photorealistic style transfer aims at changing style of a photo to that of a reference photo. For a faithful stylization, content of the photo should remain the same. Furthermore, the output photo should look like a real photo as it were captured by a camera. Figure 4.12 shows two photorealistic image stylization examples. The WCT performs well for artistic image stylization. However it generates structural artifacts (e.g., distortions on object boundaries) for photorealistic image stylization (Figure 4.14(c)). The proposed PhotoWCT is designed to suppress these structural artifacts.

4.4.1 PhotoWCT

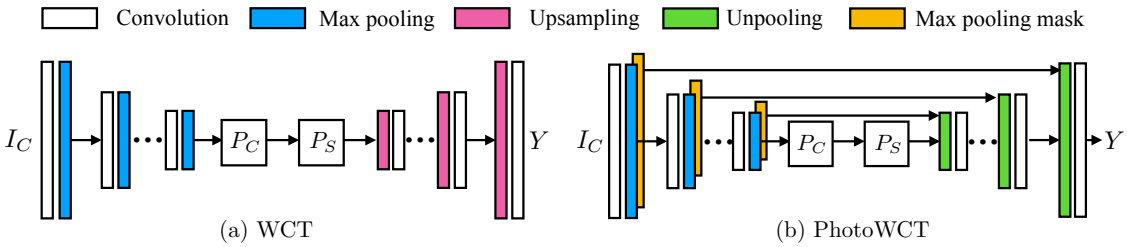
Our PhotoWCT design is motivated by the observation that the max-pooling operation in the WCT reduces spatial information in feature maps. Simply upsampling feature maps in the decoder fails to recover detailed structures of the input image. That is, we need

(a) Style (b) Content (c) Gatys *et al.* [36] (d) Luan *et al.* [80] (e) Ours

Figure 4.12: Given a style photo (a) and a content photo (b), photorealistic image stylization aims at transferring style of the style photo to the content photo as shown in (c), (d) and (e). Comparing with existing methods [36, 80], the output photos computed by our method are stylized more consistently and with fewer artifacts. Moreover, our method runs an order of magnitude faster.

to pass the lost spatial information to the decoder to facilitate reconstructing these fine details. Inspired by the success of the unpooling layer [150, 145, 86] in preserving spatial information, the PhotoWCT replaces the upsampling layers in the WCT with unpooling layers. Figure 4.13 illustrates the network architecture difference between the WCT and the proposed PhotoWCT.

Figure 4.14(c) and (d) compare the stylization results of the WCT and PhotoWCT. As highlighted in close-ups, the straight lines along the building boundary in the content image becomes zigzagged in the WCT stylization result but remains straight in the PhotoWCT result. The PhotoWCT-stylized image has much fewer structural artifacts. We also perform a user study in the experiment section to quantitatively verify that the PhotoWCT generally leads to better stylization effects than the WCT.



(a) WCT (b) PhotoWCT

Figure 4.13: The PhotoWCT and WCT share the same encoder architecture and projection steps. In the PhotoWCT, we replace the upsampling layers (pink) with unpooling layers (green). Note that the unpooling layer is used together with the pooling mask (yellow) which records *where* carries the *maximum* over each max pooling region in the corresponding pooling layer [150].

4.4.2 Photorealistic Smoothing

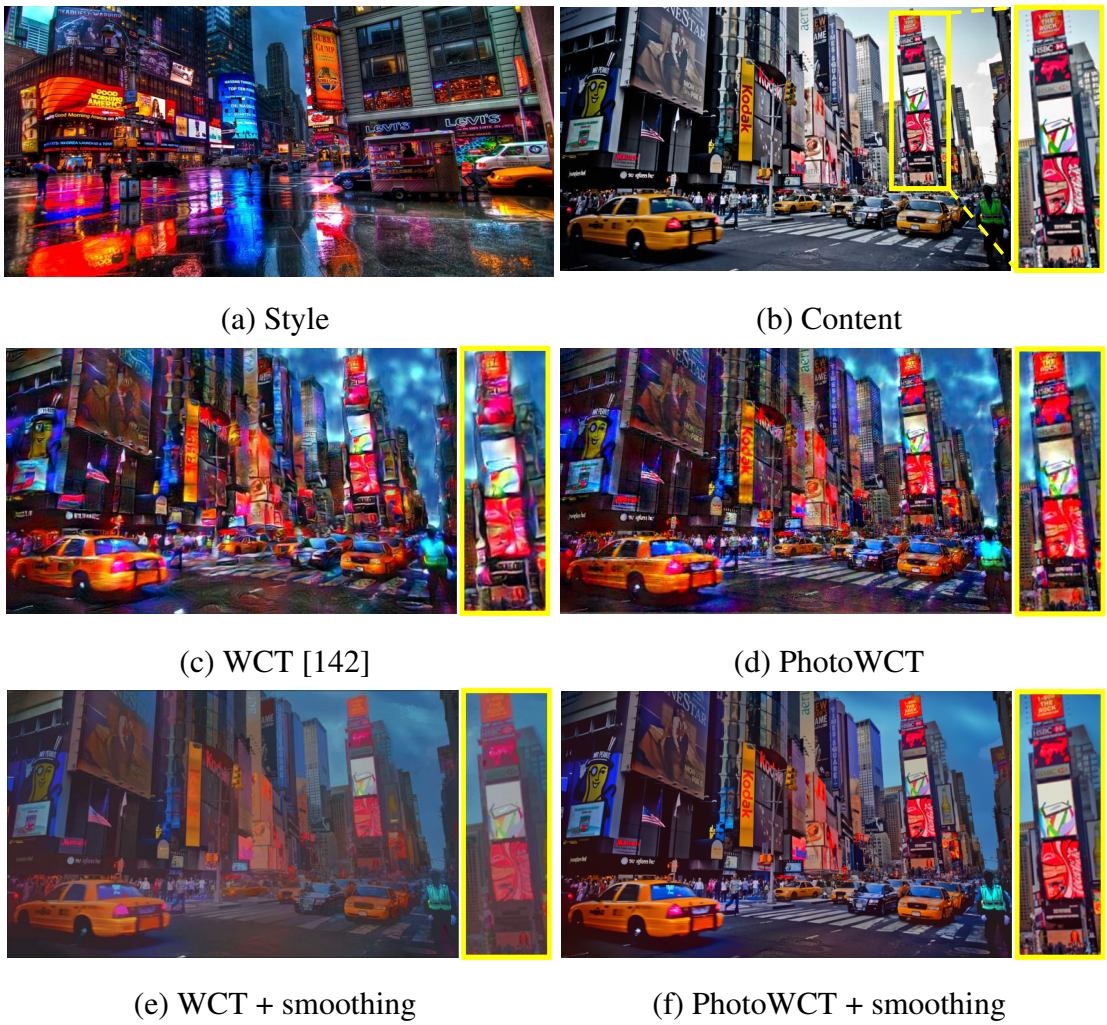
The PhotoWCT-stylized result (Figure 4.14(d)) still looks less like a photo since semantically similar regions are often stylized inconsistently. As shown in Figure 4.14, when applying the PhotoWCT to stylize the day-time photo using the night-time photo, the stylized sky region would be more photorealistic if it were uniformly dark blue instead of partly dark and partly light blue. It is based on this observation, we employ the pixel affinities in the content photo to smooth the PhotoWCT-stylized result.

We aim to achieve two goals in the smoothing step. First, pixels with similar content in a local neighborhood should be stylized similarly. Second, the output should not deviate significantly from the PhotoWCT result in order to maintain the global stylization effects. We first represent all pixels as nodes in a graph and define an affinity matrix $W = \{w_{ij}\} \in \mathbb{R}^{N \times N}$ (N is the number of pixels) to describe pixel similarities. We define a smoothness term and a fitting term that model these two goals in the following optimization problem:

$$\underset{r}{\operatorname{argmin}} \frac{1}{2}\Big(\sum_{i,j=1}^{N} w_{ij} \Big\| \frac{r_i}{\sqrt{d_{ii}}} - \frac{r_j}{\sqrt{d_{jj}}} \Big\|^2 + \lambda \sum_{i=1}^{N} \|r_i - y_i\|^2\Big), \tag{4.5}$$

where y_i is the pixel color in the PhotoWCT-stylized result Y and r_i is the pixel color in

the desired smoothed output R. The variable $d_{ii} = \sum_j w_{ij}$ is the diagonal element in the degree matrix D of W, i.e., $D = diag\{d_{11}, d_{22}, ..., d_{NN}\}$. In (4.5), λ controls the balance of the two terms.



(a) Style (b) Content

(c) WCT [142] (d) PhotoWCT

(e) WCT + smoothing (f) PhotoWCT + smoothing

Figure 4.14: The stylization output generated by the PhotoWCT better preserves local structures in the content images, which is important for the image smoothing step as shown in (e) and (f).

Our formulation is motivated by the graph-based ranking algorithms [151, 138]. In the

(a) Style (b) Content (c) PhotoWCT (Ours)

(d) MattingAff (e) GaussianAff $\sigma = 1$ (f) GaussianAff $\sigma = 0.1$

Figure 4.15: Smoothing with different affinities. To refine the PhotoWCT result in (c), it is hard to find an optimal σ for the Gaussian Affinity that performs globally well as shown in (e)-(f). In contrast, using the Matting Affinity can simultaneously smooth different regions well as shown in (d).

ranking algorithms, Y is a binary input where each element indicates if a specific item is a query ($y_i = 1$ if y_i is a query and $y_i = 0$ otherwise). The optimal solution R is the ranking values of all the items based on their pairwise affinities. In our method, we set Y as the PhotoWCT-stylized result. The optimal solution R is the smoothed version of Y based on the pairwise pixel affinities, which encourages consistent stylization within semantically similar regions. The above optimization problem is a simple quadratic problem with a closed-form solution, which is given by

$$R^* = (1 - \alpha)(I - \alpha S)^{-1} Y, \tag{4.6}$$

where I is the identity matrix, $\alpha = \frac{1}{1+\lambda}$ and S is the normalized Laplacian matrix com-

puted from I_C, i.e., $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$. As the constructed graph is often sparsely connected (i.e., most elements in W are zero), the inverse operation in (4.6) can be computed efficiently. With the closed-form solution, the smoothing step can be written as a function mapping given by:

$$R^* = \mathcal{F}_2(Y, I_C) = (1 - \alpha)(I - \alpha S)^{-1} Y. \tag{4.7}$$

Affinity. The affinity matrix W is computed using the content photo based on an 8-connected image graph assumption. While several choices of affinity metrics exist, a popular one is to define the affinity (denoted as GaussianAff) as $w_{ij} = e^{-\|I_i - I_j\|^2/\sigma^2}$ where I_i, I_j are the RGB values of adjacent pixels i, j and σ is a global scaling hyper-parameter [101]. However, it is difficult to determine the σ value in practice. It often results in either over-smoothing the entire photo (Figure 4.15(e)) or stylizing the photo inconsistently (Figure 4.15(f)). To avoid selecting one global scaling hyper-parameter, we resort to the matting affinity [68, 146] (denoted as MattingAff) where the affinity between two pixels is based on means and variances of pixels in a local window. Figure 4.15(d) shows that the matting affinity is able to simultaneously smooth different regions well.

WCT plus Smoothing. We note that the smoothing step can also remove structural artifacts in the WCT as shown in Figure 4.14(e). However, it leads to unsatisfactory stylization. The main reason is that the content photo and the WCT result are severely misaligned due to spatial distortions. For example, a stylized pixel of the building in the WCT result may correspond to a pixel of the sky in the content photo. Consequently this causes wrong queries in Y for the smoothing step. This shows why we need to use the PhotoWCT to remove distortions first. Figure 4.14(f) shows that the combination of PhotoWCT and smoothing leads to better photorealism while still maintaining faithful stylization.

4.4.3 Results

In the section, we will first discuss the implementation details. We will then present visual and user study evaluation results. Finally, we will analyze various design choices and run-time of the proposed algorithm.

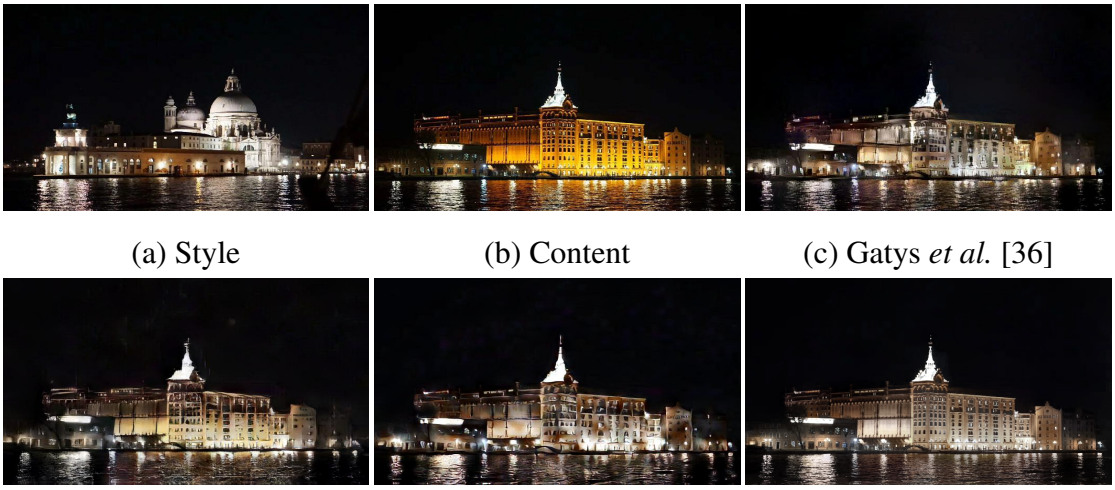
Implementation details. We use the layers from $conv1\_1$ to $conv4\_1$ in the VGG-19 network [105] for the encoder $\mathcal{E}$. The encoder weights are given by ImageNet-pretrained weights. The decoder $\overline{\mathcal{D}}$ is the inverse of the encoder. We train the decoder by minimizing the sum of the L_2 reconstruction loss and perceptual loss [56] using the Microsoft COCO dataset [73]. We adopt the multi-level stylization strategy proposed in the WCT [142] where we apply the PhotoWCT to VGG features in different layers.

Similar to the state-of-the-art methods [37, 80], our algorithm can leverage semantic label maps for obtaining better stylization results when they are available. When performing PhotoWCT stylization, for each semantic label, we compute a pair of projection matrices P_C and P_S using the features from the image regions with the same label in the content and style photos, respectively. The pair is then used to stylize these image regions. With a semantic label map, content and style matching can be performed more accurately. We note that the proposed algorithm does not need precise semantic label maps for obtaining good stylization results. Finally, we also use the efficient filtering step described in Luan *et al.* [80] for post-processing.

Visual comparison. We compare the proposed algorithm to two categories of stylization algorithms: photorealistic and artistic. The evaluated photorealistic stylization algorithms include Reinhard *et al.* [96], Pitié *et al.* [92], and Luan *et al.* [80]. Both Reinhard *et al.* [96] and Pitié *et al.* [92] represent classical techniques that are based on color statistics matching, while Luan *et al.* [80] is based on neural style transfer [36]. On the other hand, the set of evaluated artistic stylization algorithms include Gatys *et al.*[36], Huang *et al.*[53], and the WCT [142]. They all utilize deep networks.

Figure 4.16: Visual comparisons with photorealistic stylization methods. In addition to color transfer, our method also synthesizes patterns in the style photos (e.g., the dark cloud in the top example, the snow at the bottom example).

Figure 4.16 shows visual results of the evaluated photorealistic stylization algorithms.



(a) Style (b) Content (c) Gatys *et al.* [36]

(d) Huang *et al.* [53] (e) Li *et al.* [142] (f) Ours

Figure 4.17: Visual comparison with artistic stylization algorithms. Note the structural distortions on object boundaries (e.g., building) and detailed edges (e.g., sea, cloud) generated by the competing stylization methods.

Overall, the images generated by the proposed algorithm exhibit better stylization effects. While both Reinhard *et al.* [96] and Pitié *et al.* [92] change colors of the content photos, they fail to transfer the style. We argue that photorealistic stylization cannot be purely achieved via color transfer. It requires adding new patterns that represent the style photo to the content photo. For example, in the third example of Figure 4.16 (bottom), our algorithm not only changes the color of ground regions to white but also synthesizes the snow patterns as they appear in the style photo. The method of Luan *et al.* [80] achieves good stylization effects at first glance. However, a closer look reveals that the generated photos contain noticeable artifacts, e.g., the irregular brightness on buildings and trees. Several semantically similar regions are stylized inconsistently.

Figure 4.17 shows the visual comparison between the proposed algorithm and artistic stylization algorithms. Although the other evaluated algorithms are able to transfer the style well, they render noticeable structural artifacts and inconsistent stylizations across

the images. In contrast, our method produces more photorealistic results.

User studies. We resort to user studies for performance evaluation since photorealistic image stylization is a highly subjective task. Our benchmark dataset consists of a set of 25 content–style pairs provided by Luan *et al.* [80]. We use the Amazon Mechanical Turk (AMT) platform for evaluation. In each question, we show the AMT workers a content–style pair and the stylized results from the evaluated algorithms displayed in random order. The AMT workers (lifetime Human Intelligent Task approval rate greater than 98%) are asked to select a stylized result based on the instructions. Each question is answered by 10 different workers. Hence, the performance score for each study is computed based on 250 questions. We compute the average number of times the images from an algorithm is selected, which is used as the preference score of the algorithm.

We conduct two user studies. In one study, we ask the AMT workers to select which stylized photo better carries the target style. In the other study, we ask the workers to select which stylized photo looks more like a real photo (containing fewer artifacts). Through the studies, we would like to answer which algorithm better stylizes content images and which renders better photorealistic outputs.

In Table 4.3, we compare the proposed algorithm to Luan *et al.* [80], which is the current state-of-the-art. The results show that 63.1% of the users prefer the stylization results generated by our algorithm and 73.5% regard our output photos as more photorealistic. We also compare our algorithm to the classical algorithm of Pitié *et al.* [92]. From Table 4.3, our results are as photorealistic as those computed by the classical algorithm (which simply performs color matching), and 55.2% of the users consider our stylization results better.

Table 4.4 compares our algorithm with the artistic stylization algorithms for user preference scores. We find our algorithm achieves a score of 56.4% and 65.6% for the stylization effect and photorealism, which are significantly better than the other algorithms. The artistic stylization algorithms do not perform well since they are not designed for the

Table 4.3: User preference: proposed vs. Luan *et al.* and proposed vs. Pitié *et al.*

	Luan *et al.* [80] / proposed	Pitié *et al.* [92] / proposed
Better stylization	36.9% / **63.1%**	44.8% / **55.2%**
Fewer artifacts	26.5% / **73.5%**	48.8% / **51.2%**

Table 4.4: User preference: proposed versus *artistic* stylization algorithms.

	Gatys *et al.* [36]	Huang *et al.* [53]	Li *et al.* [142]	proposed
Better stylization	19.2%	8.4%	16.0%	**56.4%**
Fewer artifacts	21.6%	6.0%	6.8%	**65.6%**

photorealistic stylization task.

WCT versus PhotoWCT. We compare the proposed algorithm with a variant where the PhotoWCT step is replaced by the WCT [142]. Again, we conduct two user studies on stylization effects and photorealism as described earlier. The result shows that the proposed algorithm is favored over its variant for better stylization 83.6% of the times and favored for better photorealism 83.2% of the times.

Sensitivity analysis on λ. In the photorealistic smoothing step, the λ balances between the smoothness term and fitting term in (4.5). A smaller λ renders smoother results, while a larger λ renders results that are more faithful to the queries (the PhotoWCT result). Figure 4.18 shows results of using different λ values. In general, decreasing λ helps remove artifacts and hence improves photorealism. However, if λ is too small, the output image tends to be over-smoothed. In order to find the optimal λ, we perform a grid search. We use the similarity between the boundary maps extracted from stylized and original content photos as the criteria since object boundaries should remain the same despite the stylization [17]. We employ the HED method [133] for boundary detection and use two standard

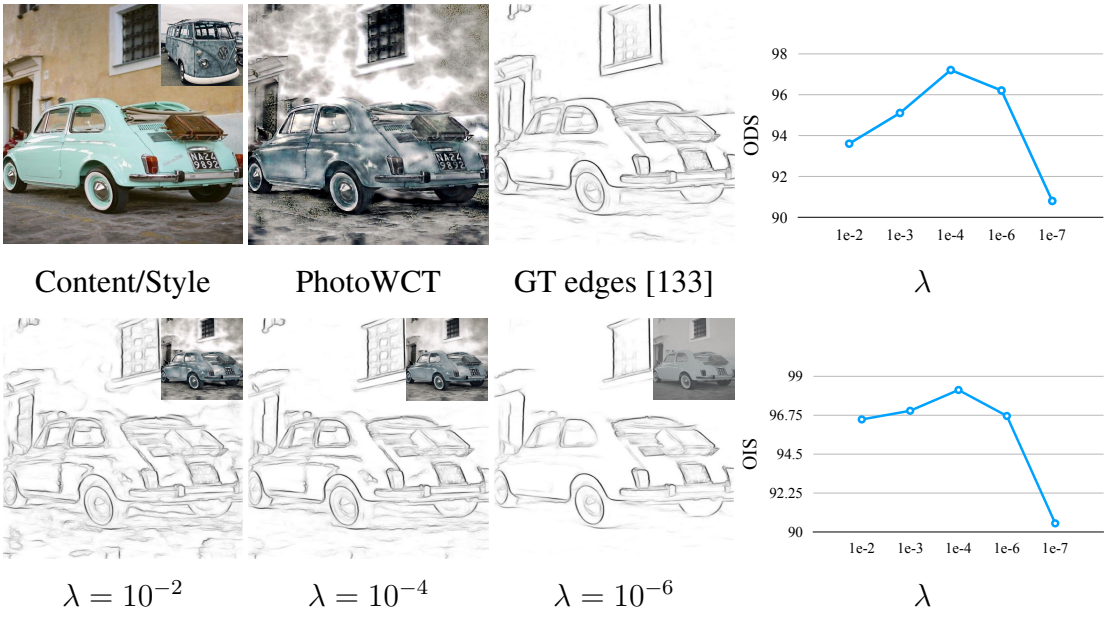


Figure 4.18: Visualization of effects of using different λ values in the photorealistic smoothing step. We show the edge maps of different stylization results (inset) at bottom and compare them with the edge map of the content in terms of the ODS and OIS metric (rightmost).

boundary detection metrics: ODS and OIS. A higher ODS or OIS score means a stylized photo better preserves the content in the original photo. The average scores over the benchmark dataset are shown on the rightmost of Figure 4.18. Based on the results, we use $\lambda = 10^{-4}$ in all the experiments.

Alternative smoothing techniques. In Figure 4.19, we compare our photorealistic smoothing step with two alternative approaches. In the first approach, we use the PhotoWCT-stylized photo as the initial solution for solving the second optimization problem in the method of Luan *et al.* [80]. The result is shown in Figure 4.19(b). This approach leads to noticeable artifacts as the road color is distorted. In the second approach, we use the method of Mechrez *et al.* [84], which refines stylized results by matching the gradients

(a) PhotoWCT (b) Luan *et al.* [80] (c) Mechrez *et al.* [84] (d) proposed

Figure 4.19: Comparison between using our photorealistic smoothing step and other refinement methods (b)-(d).

in the output photo to those in the content photo. As shown in Figure 4.19(c), we find this approach performs well for removing structural distortions on boundaries but does not remove visual artifacts. In contrast, our method (Figure 4.19(d)) generates more photorealistic results with an efficient closed-form solution.

Run-time. In Table 4.5, we compare the run-time of the proposed algorithm to that of the state-of-the-art [80]. We note that while our algorithm has a closed-form solution, Luan *et al.* [80] rely on non-convex optimization. To stylize a photo, Luan *et al.* [80] solve two non-convex optimization problems sequentially where the solution to the first optimization problem is used as an initial solution to solve the second optimization problem. We report the total run-time required for obtaining the final stylization results. We resize the content images in the benchmark dataset to different sizes and report the average run-time for each image size. The experiment is conducted on a PC with an NVIDIA Titan X Pascal GPU. To stylize images of 1024×512 resolution, our algorithm takes 13.16 seconds, which is 49 times faster than 650.45 seconds achieved by Luan *et al.* [80].

In Table 4.5, we also report the run-time of each step in our algorithm. We find the smoothing step takes most of the computation time, since it involves inverting the sparse matrix W in (4.6) using the LU decomposition. By employing efficient LU-decomposition algorithms developed for large sparse matrices, the complexity can be roughly determined by the number of non-zero entries in the matrices only. In our case, since each pixel is

Table 4.5: Run-time comparison. We compute the average run time (in seconds) of the evaluated algorithms across various image resolutions.

Image resolution	Luan *et al.*[80]	proposed	PhotoWCT	smoothing	`approx`
256×128	79.61	0.96	0.40	0.56	0.41
512×256	186.52	2.95	0.42	2.53	0.47
768×384	380.82	7.05	0.53	6.52	0.55
1024×512	650.45	13.16	0.56	12.60	0.64

Table 4.6: User preference score comparison: comparing `approx` (the fast approximation of the proposed algorithm) to the proposed algorithm as well as other photorealistic stylization algorithms.

	proposed/`approx`	Luan *et al.* [80]/`approx`	Pitié *et al.* [92]/`approx`
Better stylization	**59.6%** / 40.4	36.4 / **63.6%**	46.0 / **54.0%**
Fewer artifacts	**52.8%** / 47.2	20.8 / **79.2%**	46.8 / **53.2%**

only connected to its neighbors (e.g., 3×3 window), the number of non-zero values in W grows linearly with the image size.

For further speed-up, we can approximate the smoothing step using guided image filtering [46], which can smooth the PhotoWCT output based on the content photo. We will refer to this version of our algorithm `approx`. Although approximating the smoothing step with guided image filtering results in slightly degraded performance as comparing to the original algorithm, it leads to a large speed gain as shown in Table 4.5. To stylize images of 1024×512 resolution, `approx` only takes 0.64 seconds, which is 1,016 times faster than 650.45 seconds achieved by Luan *et al.* [80]. To quantify the performance degradation due to the approximation, we conduct additional user studies comparing the proposed algorithm and its approximation. We use the same evaluation protocol as described above. The results are shown in Table 4.6. In general, the stylization results

rendered by `approx` are less preferred by the users as compared to those generated by the full algorithm. However, the results from `approx` are still preferred over other methods in terms of both stylization effects and photorealism.

4.5 Concluding Remarks

In this work, we propose a universal style transfer algorithm that does not require learning for each individual style. By unfolding the image generation process via training an auto-encoder for image reconstruction, we integrate the whitening and coloring transforms in the feed-forward passes to match the statistical distributions and correlations between the intermediate features of content and style. We also present a multi-level stylization pipeline, which takes all level of information of a style into account, for improved results. In addition, the proposed approach is shown to be equally effective for texture synthesis. Experimental results demonstrate that the proposed algorithm achieves favorable performance against the state-of-the-art methods in generalizing to arbitrary styles.

Chapter 5

Generative Face Completion

In this work, we propose an effective face completion algorithm using a deep generative model. Different from well-studied background completion, the face completion task is more challenging as it often requires to generate semantically new pixels for the missing key components (e.g., eyes and mouths) that contain large appearance variations. Unlike existing nonparametric algorithms that search for patches to synthesize, our algorithm directly generates contents for missing regions based on a neural network. The model is trained with a combination of a reconstruction loss, two adversarial losses and a semantic parsing loss, which ensures pixel faithfulness and local-global contents consistency. With extensive experimental results, we demonstrate qualitatively and quantitatively that our model is able to deal with a large area of missing pixels in arbitrary shapes and generate realistic face completion results.

5.1 Introduction

Image completion, as a common image editing operation, aims to fill the missing or masked regions in images with plausibly synthesized contents. The generated contents can either be as accurate as the original, or simply fit well within the context such that the completed image appears to be visually realistic. Most existing completion algorithms [4, 52] rely on low-level cues to search for patches from known regions of the same image and synthesize the contents that locally appear similarly to the matched patches. These approaches are all fundamentally constrained to copy existing patterns and structures from the known regions. The copy-and-paste strategy performs particularly well for background completion (e.g., grass, sky, and mountain) by removing foreground objects and filling the unknown regions with similar pattens from backgrounds.

However, the assumption of similar patterns can be found in the same image does not hold for filling missing parts of an object image (e.g., face). Many object parts contain unique patterns, which cannot be matched to other patches within the input image, as shown in Figure 5.1(b). An alternative is to use external databases as references [45]. Although similar patches or images may be found, the unique patterns of objects that involve semantic representation are not well modeled, since both low-level [4] and mid-level [52] visual cues of the known regions are not sufficient to infer semantically valid contents in missing regions.

In this work, we propose an effective object completion algorithm using a deep generative model. The input is first masked with noise pixels on randomly selected square region, and then fed into an autoencoder [121]. While the encoder maps the masked input to hidden representations, the decoder generates a filled image as its output. We regularize the training process of the generative model by introducing two adversarial losses [42]: a local loss for the missing region to ensure the generated contents are semantically coherent, and a global one for the entire image to render more realistic and visually pleasing results. In addition, we also propose a face parsing network [75, 106, 66] as an additional loss to

(a) (b) (c)

Figure 5.1: Face completion results. In each row from left to right: (a) original image (128 $\times$ 128 pixels). (b) masked input. (c) completion results by our method. In the top row, the face is masked by a square. In the bottom row we show a real example where the mouth region is occluded by the microphone.

regularize the generation procedure and enforce a more reasonable and consistent result with contexts. This generative model allows fast feed-forward image completion without requiring an external databases as reference. For concreteness, we apply the proposed object completion algorithm on face images.

The main contributions of this work are summarized as follows. First, we propose a deep generative completion model that consists of an encoding-decoding generator and two adversarial discriminators to synthesize the missing contents from random noise. Second, we tackle the challenging face completion task and show the proposed model is able to generate semantically valid patterns based on learned representations of this object class. Third, we demonstrate the effectiveness of semantic parsing in generation, which renders

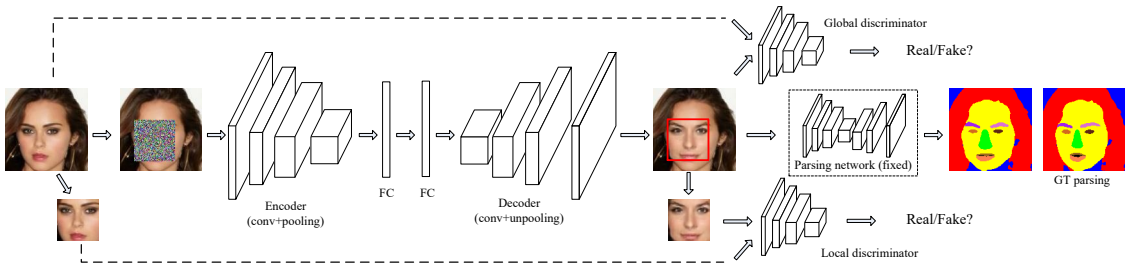


Figure 5.2: Network architecture. It consists of one generator, two discriminators and a parsing network. The generator takes the masked image as input and outputs the generated image. We replace pixels in the non-mask region of the generated image with original pixels. Two discriminators are learned to distinguish the synthesize contents in the mask and whole generated image as real and fake. The parsing network, which is a pretrained model and remains fixed, is to further ensure the new generated contents more photo-realistic and encourage consistency between new and old pixels. Note that only the generator is needed during the testing.

the completion results that look both more plausible and consistent with surrounding contexts.

5.2 Proposed Algorithm

In this section, we describe the proposed model for object completion. Given a masked image, our goal is to synthesize the missing contents that are both semantically consistent with the whole object and visually realistic. Figure 5.2 shows the proposed network that consists of one generator, two discriminators, and a parsing network.

5.2.1 Generator

The generator $\mathcal{G}$ is designed as an autoencoder to construct new contents given input images with missing regions. The masked (or corrupted) input, along with the filled noise, is first mapped to hidden representations through the encoder. Unlike the original GAN model [42] which directly starts from a noise vector, the hidden representations obtained from the encoder capture more variations and relationships between unknown and known regions, which are then fed into the decoder for generating contents.

We use the architecture from "conv1" to "pool3" of the VGG-19 [105] network, stack two more convolution layers and one more pooling layer on top of that, and add a fully-connected layer after that as the encoder. The decoder is symmetric to the encoder with unpooling layers.

5.2.2 Discriminator

The generator can be trained to fill the masked region or missing pixels with small reconstruction errors. However, it does not ensure that the filled region is visually realistic and coherent. As shown in Figure 5.3(c), the generated pixels are quite blurry and only capture the coarse shape of missing face components. To encourage more photo-realistic results, we adopt a discriminator $\mathcal{D}$ that serves as a binary classifier to distinguish between real and fake images. The goal of this discriminator is to help improve the quality of synthesized results such that the trained discriminator is fooled by unrealistic images.

We first propose a local $\mathcal{D}$ for the missing region which determines whether the synthesized contents in the missing region are real or not. Compared with Figure 5.3(c), the network with local $\mathcal{D}$ (shown in Figure 5.3(d)) begins to help generate details of missing contents with sharper boundaries. It encourages the generated object parts to be semantically valid. However, its limitations are also obvious due to the locality. First, the local loss can neither regularize the global structure of a face, nor guarantee the statistical consistency within and outside the masked regions. Second, while the generated new pixels are

(a) Original (b) Input (c) M1 (d) M2 (e) M3 (f) M4 (g) M5

Figure 5.3: Completion results under different settings of our model. (c) M1: L_r. (d) M2: $L_r + L_{a_1}$. (e) M3: $L_r + L_{a_1} + L_{a_2}$. (f) M4: $L_r + L_{a_1} + L_{a_2} + L_p$. The result in (f) shows the most realistic and plausible completed content. It can be further improved through post-processing techniques such as (g) M5: M4 + Poisson blending [90] to eliminate subtle color difference along mask boundaries.

conditioned on their surrounding contexts, a local $\mathcal{D}$ can hardly generate a direct impact outside the masked regions during the back propagation, due to the unpooling structure of the decoder. Consequently, the inconsistency of pixel values along region boundaries is obvious.

Therefore, we introduce another global $\mathcal{D}$ to determine the faithfulness of an entire image. The fundamental idea is that the newly generated contents should not only be realistic, but also consistent to the surrounding contexts. From Figure 5.3(e), the network with additional global $\mathcal{D}$ greatly alleviates the inconsistent issue and further enforce the generated contents to be more realistic. We note that the architecture of two discriminators are similar to [93].

5.2.3 Semantic Regularization

With a generator and two discriminators, our model can be regarded as a variation of the original GAN [42] model that is conditioned on contexts (e.g., non-mask regions).

However as a bottleneck, the GAN model tends to generate independent facial components that are likely not suitable to the original subjects with respect to facial expressions and parts shapes, as shown in Figure 5.3(e). The top one is with big weird eyes and the bottom one contains two asymmetric eyes. Furthermore, we find the global $\mathcal{D}$ is not effective in ensuring the consistency of fine details in the generated image. For example, if only one eye is masked, the generated eye does not fit well with another unmasked one. We show another two examples in Figure 5.4(c) where the generated eye is obviously asymmetric to the unmasked one although the generated eye itself is already realistic. Both cases indicate that more regularization is needed to encourage the generated faces to have similar high-level distributions with the real faces.

Therefore we introduce a semantic parsing network to further enhance the harmony of the generated contents and existing pixels. The parsing network is an autoencoder which bears some resemblance to the semantic segmentation method [139]. The parsing result of the generated image is compared with the one of the original image. As such, the generator is forced to learn where to generate features with more natural shape and size. In Figure 5.3(e)-(f) and Figure 5.4(c)-(d), we show the generated images between models without and with the smenatic regularization.

5.2.4 Objective Function

We first introduce a reconstruction loss L_r to the generator, which is the L_2 distance between the network output and the original image. With the L_r only, the generated contents tend to be blurry and smooth as shown in Figure 5.3(c). The reason is that since the L_2 loss penalizes outliers heavily, and the network is encouraged to smooth across various hypotheses to avoid large penalties.

By using two discriminators, we employ the adversarial loss which is a reflection of how the generator can maximally fool the discriminator and how well the discriminator

(a) original (b) masked input (c) w/o parsing (d) w/ parsing

Figure 5.4: Comparison between the result of models without and with the parsing regularization.

can distinguish between real and fake. It is defined as

$$L_{a_i} = \min_{\mathcal{G}} \max_{\mathcal{D}} \ \mathcal{E}_{x\sim p_{data}(x)}[\log \mathcal{D}(x)] + \mathcal{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))], \tag{5.1}$$

where $p_{data}(x)$ and $p_z(z)$ represent the distributions of noise variables z and real data x. The two discriminative networks $\{a_1, a_2\}$ share the same definition of the loss function. The only difference is that the local discriminator only provides training signals (loss gradients) for the missing region while the global discriminator back-propagates loss gradients across the entire image.

In the parsing network, the loss L_p is the simple pixel-wise softmax loss [78, 139]. The overall loss function is defined by

$$L = L_r + \lambda_1 L_{a_1} + \lambda_2 L_{a_2} + \lambda_3 L_p, \tag{5.2}$$

where λ_l, λ_2 and λ_3 are the weights to balance the effects of different losses.

5.2.5 Training Neural Networks

To effectively train our network, we use the curriculum strategy [6] by gradually increasing the difficulty level and network scale. The training process is scheduled in three stages. First, we train the network using the reconstruction loss to obtain blurry contents. Second, we fine-tune the network with the local adversarial loss. The global adversarial loss and semantic regularization are incorporated at the last stage, as shown in Figure 5.3. Each stage prepares features for the next one to improve, and hence greatly increases the effectiveness and efficiency of network training. For example, in Figure 5.3, the reconstruction stage (c) restores the rough shape of the missing eye although the contents are blurry. Then local adversarial stage (d) then generates more details to make the eye region visually realistic, and the global adversarial stage (e) refines the whole image to ensure that the appearance is consist around the boundary of the mask. The semantic regularization (f) finally further enforces more consistency between components and let the generated result to be closer to the actual face. When training with the adversarial loss, we use a method similar to [93] especially to avoid the case when the discriminator is too strong at the beginning of the training process.

5.3 Experimental Results

We carry out extensive experiments to demonstrate the ability of our model to synthesize the missing contents on face images. The hyper-parameters (e.g., learning rate) for the network training are set as suggested in [128]. To balance the effects of different losses, we use $\lambda_l = 300$, $\lambda_2 = 300$ and $\lambda_3 = 0.005$ in our experiments.

5.3.1 Datasets

We use the CelebA [76] dataset to learn and evaluate our model. It consists of 202,599 face images and each face image is cropped, roughly aligned by the position of two eyes,

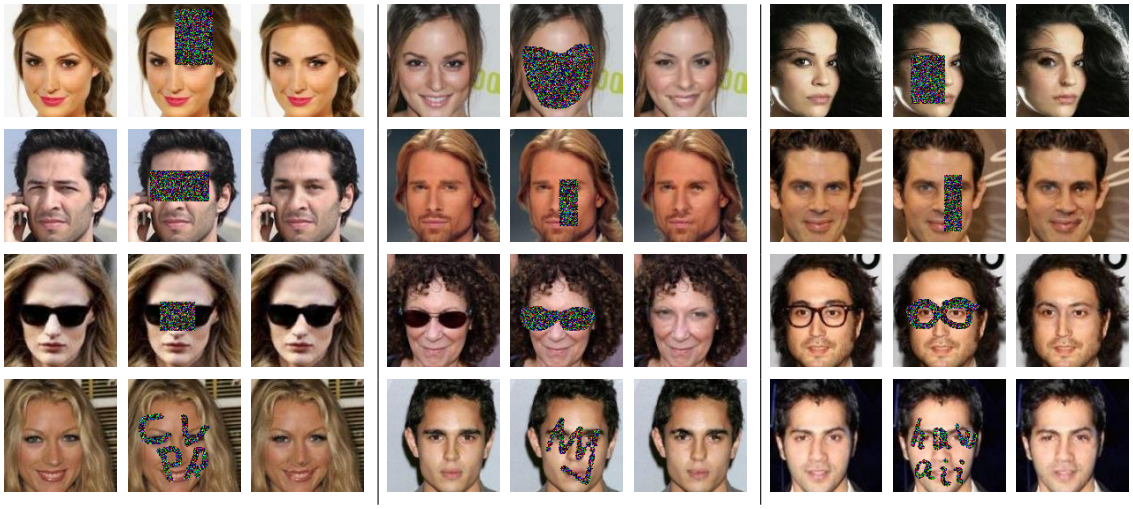


Figure 5.5: Face completion results on the CelebA [76] test dataset. In each panel from left to right: original images, masked inputs, our completion results.

and rescaled to $128 \times 128 \times 3$ pixels. We follow the standard split with 162,770 images for training, 19,867 for validation and 19,962 for testing. We set the mask size as 64×64 for training to guarantee that at least one essential facial component is missing. If the mask only covers smooth regions with a small mask size, it will not drive the model to learn semantic representations. To avoid over-fitting, we do data augmentation that includes flipping, shift, rotation (+/- 15 degrees) and scaling. During the training process, the size of the mask is fixed but the position is randomly selected. As such, the model is forced to learn the whole object in an holistic manner instead of a certain part only.

5.3.2 Face Parsing

Since face images in the CelebA [76] dataset do not have segment labels, we use the Helen face dataset [66] to train a face parsing network for regularization. The Helen dataset consists of 2,330 images and each face has 11 segment labels covering every main component of the face (e.g., hair, eyebrows, eyes) labelled by [106]. We roughly crop the

face in each image with the size of 128×128 first and then feed it into the parsing network to predict the label for each pixel. Our parsing network bears some resemblance to the semantic segmentation method [139] and we mainly modify its last layer with 11 outputs. We use the standard training/testing split and obtain a parsing model, which achieves the f-score of 0.851 with overall facial components on the Helen test dataset, compared to the state-of-the-art multi-objective based model [75], with the corresponding f-score of 0.854. This model can be further improved with more careful hyperparameter tuning but is currently sufficient to improve the quality of face completion.

Once the parsing network is trained, it remains fixed in our generation framework. We first use the network on the CelebA training set to obtain the parsing results of originally unmasked faces as the ground truth, and compare them with the parsing on generated faces during training. The parsing loss is eventually back-propagated to the generator to regularize face completion. The proposed semantic regularization can be regarded as measuring the distance in feature space where the sensitivity to local image statistics can be achieved [25].

5.3.3 Face Completion

Qualitative results. Figure 5.5 shows our face completion results on the CelebA test dataset. In each test image, the mask covers at least one key facial components. The third column of each panel shows our completion results are visually realistic and pleasing. Note that during the testing, the mask does not need to be restricted as a 64×64 square mask, but the number of total masked pixels is suggested to be no more than 64×64 pixels. We show typical examples with one big mask covering at least two face components (e.g., eyes, mouths, eyebrows, hair, noses) in the first two rows. We specifically present more results on eye regions since they can better reflect how realistic of the newly generated faces are, with the proposed algorithm. Overall, the algorithm can successfully complete the images with faces in side views, or partially/completely corrupted by the masks with

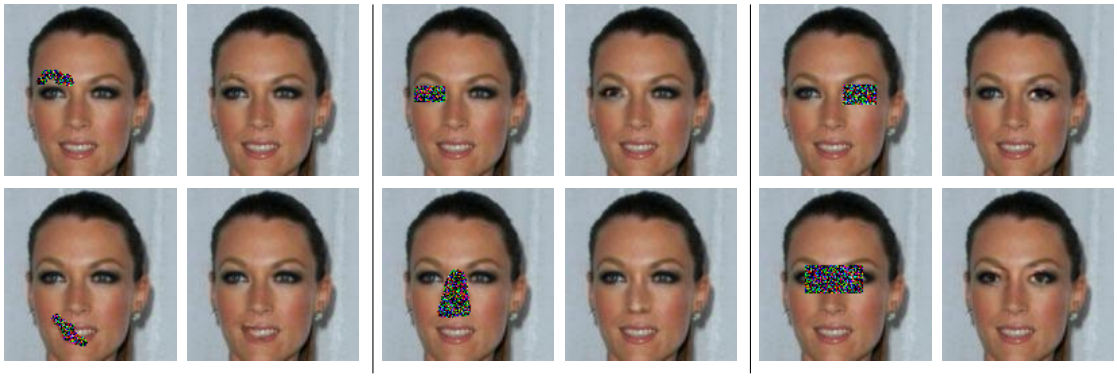


Figure 5.6: Face part completion. In each panel, left: masked input, right: our completion result.

different shapes and sizes.

We present a few examples in the third row where the real occlusion (e.g., wearing glasses) occurs. As sometimes whether a region in the image is occluded or not is subjective, we give this option for users to assign the occluded regions through drawing masks. The results clearly show that our model is able to restore the partially masked eyeglasses, or remove the whole eyeglasses or just the frames by filling in realistic eyes and eyebrows.

In the last row, we present examples with multiple, randomly drawn masks, which are closer to real-world application scenarios. Figure 5.6 presents completion results where different key parts (e.g., eyes, nose, and mouth) of the same input face image are masked. It shows that our completion results are consistent and realistic regardless of the mask shapes and locations.

Quantitative results. In addition to the visual results, we also perform quantitative evaluation using three metrics on the CelebA test dataset (19,962 images). The first one is the peak signal-to-noise ratio (PSNR) which directly measures the difference in pixel values. The second one is the structural similarity index (SSIM) that estimates the holistic similarity between two images. Lastly we use the identity distance measured by the OpenFace toolbox [2] to determine the high-level semantic similarity of two faces. These three met-

(a) O1 (b) O2 (c) O3 (d) O4 (e) O5 (f) O6

Figure 5.7: Simulate face occlusions happened in real scenario with different masks O1-O6. From left to right: left half, right half, two eyes, left eye, right eye, and lower half.

rics are computed between the completion results obtained by different methods and the original face images. The results are shown in Table 5.1-5.3. Specifically, the stepwise contribution of each component is shown from the 2nd to the 5th column of each table, where M1-M5 correspond to five different settings of our own model in Figure 5.3 and O1-O6 are six different masks for evaluation as shown in Figure 5.7.

We then compare our model with the ContextEncoder [89] (CE). Since the CE model is originally not trained for faces, we retrain the CE model on the CelebA dataset for fair comparisons. As the evaluated masks O1-O6 are not in the image center, we use the *inpaintRandom* version of their code and mask 25% pixels masked in each image. Finally we also replace the non-mask region of the output with original pixels. The comparison between our model (M4) and CE in 5th and 6th column show that our model performs generally better than the CE model, especially on large masks (e.g., O1-O3, O6). In the last column, we show that the poisson blending [90] can further improve the performance.

Note that we obtain relatively higher PSNR and SSIM values when using the reconstruction loss (M1) only but it does not imply better qualitative results, as shown in Figure 5.3(c). These two metrics simply favor smooth and blurry results. We note that the model M1 performs poorly as it hardly recovers anything and is unlikely to preserve the identity well, as shown in Table 5.3.

Table 5.1: Quantitative evaluations in terms of SSIM at six different masks O1-O6. Higher values are better.

	M1	M2	M3	M4	CE	M5
O1	0.798	0.753	0.782	0.804	0.772	**0.824**
O2	0.805	0.763	0.787	0.808	0.774	**0.826**
O3	0.723	0.675	0.708	0.731	0.719	**0.759**
O4	0.747	0.701	0.741	0.759	0.754	**0.789**
O5	0.751	0.706	0.732	0.755	0.757	**0.784**
O6	0.807	0.764	0.808	0.824	0.818	**0.841**

Table 5.2: Quantitative evaluations in terms of PSNR at six different masks O1-O6. Higher values are better.

	M1	M2	M3	M4	CE	M5
O1	18.9	17.8	18.9	19.4	18.6	**20.0**
O2	18.7	17.9	18.7	19.3	18.4	**19.8**
O3	17.9	17.2	17.7	18.3	17.9	**18.8**
O4	18.6	17.7	18.5	19.1	19.0	**19.7**
O5	18.7	17.6	18.4	18.9	19.1	**19.5**
O6	18.8	17.3	19.0	19.7	19.3	**20.2**

5.3.4 Face recognition

The identity distance in Table 5.3 partly reveals the network ability of preserving the identity information. In order to test to what extent the face identity can be preserved across its different examples, we evaluate our completion results in the task of face recognition. Note that this task simulates occluded face recognition, which is still an open problem in computer vision. Given a probe face example, the goal of recognition is to find an example from the gallery set that belongs to the same identity. We randomly split the

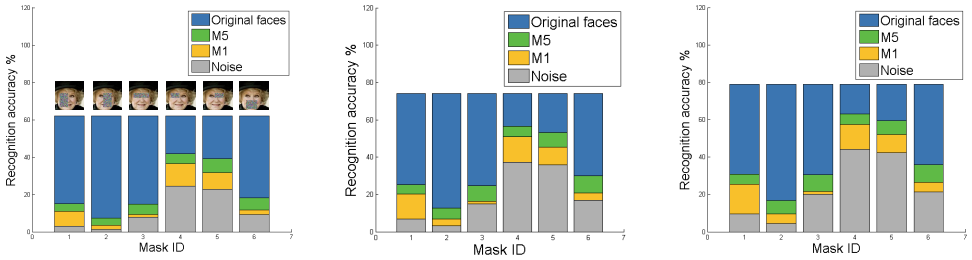
Table 5.3: Quantitative evaluations in terms of identity distance at six different masks O1-O6. Lower values are better.

	M1	M2	M3	M4	CE	M5
O1	0.763	0.775	0.694	0.602	0.701	**0.534**
O2	1.05	1.02	0.894	0.838	0.908	**0.752**
O3	0.781	0.693	0.674	0.571	0.561	**0.549**
O4	0.310	0.307	0.265	0.238	0.236	**0.212**
O5	0.344	0.321	0.297	0.256	0.251	**0.231**
O6	0.732	0.714	0.593	0.576	0.585	**0.541**

CelebA [76] test dataset into the *gallery* and *probe* set, to make sure that each identity has roughly the same amount of images in each set. Finally, we obtain the gallery and probe set with roughly 10,000 images respectively, covering about 1,000 identities.

We apply six masking types (O1-O6) for each probe image, as shown in Figure 5.7. The probe images are new faces restored by the generator. These six masking types, to some extent, simulate the occlusions that possibly occurs in real scenarios. For example, masking two eyes mainly refers to the occlusion by glasses and masking lower half face matches the case of wearing the scarf. Each completed probe image is matched against those in the gallery, and top ranked matches can be analyzed to measure recognition performance. We use the OpenFace [2] toolbox to find top K nearest matches based on the identity distance and report the average top K recognition accuracy over all probe images in Figure 5.8.

We carry out experiments with four variations of the probe image: the original one, the completed one by simply filling random noise, by our reconstruction based model M1 and by our final model M5. The recognition performance using original probe faces is regarded as the upper bound. Figure 5.8 shows that using the completed probe by our model M5 (green) achieves the closest performance to the upper bound (blue). Although there is still a large gap between the performance of our M5 based recognition and the upper



(a) Top1 (b) Top3 (c) Top5

Figure 5.8: Recognition accuracy comparisons on masked (or occluded) faces. Given a masked probe face, we first complete it and then use it to search examples of the same identity in the gallery. We report the Top1, Top3, and Top5 recognition accuracy of three different completion methods. The accuracy by using the original unmasked probe face (blue) is treated as the standard to compare.

bound, especially when the mask is large (e.g., O1, O2), the proposed algorithm makes significant improvement with the completion results compared with that by either noise filling or the reconstruction loss (L_r). We consider the identity-preserving completion to be an interesting direction to pursue.

5.3.5 Limitation

Although our model is able to generate semantically plausible and visually pleasing contents, it has some limitations. The faces in the CelebA dataset are roughly cropped and aligned [76]. We implement various data augmentation to improve the robustness of learning, but find our model still cannot handle some unaligned faces well. We show one failure case in the first row of Figure 5.9. The unpleasant synthesized contents indicate that the network does not recognize the position/orientation of the face and its corresponding components. This issue can be alleviated with 3D data augmentation.

In addition, our model does not fully exploit the spatial correlations between adjacent pixels as shown in the second row of Figure 5.9. The proposed model fails to recover the

Figure 5.9: Model limitations. Left: our model fails to generate the eye for an unaligned face. Right: it is still hard to generate the semantic part with right attributes (e.g., red lipsticks).

correct color of the lip, which is originally painted with red lipsticks. In our future work, we plan to investigate the usage of pixel-level recurrent neural network (PixelRNN [117]) to address this issue.

5.4 Conclusion

We propose a deep generative network for face completion. The network is based on a GAN, with an autoencoder as the generator, two adversarial loss functions (local and global) and a semantic regularization as the discriminators. The proposed model can successfully synthesize semantically valid and visually plausible contents for the missing facial key parts from random noise. Both qualitative and quantitative experiments show that our model generates the completion results of high perceptual quality and is quite flexible to handle a variety of maskings or occlusions (e.g., different positions, sizes, shapes).

Chapter 6

Flow-Grounded Video Prediction from Still Images

Existing video prediction methods mainly rely on observing multiple historical frames or focus on predicting the next one-frame. In this work, we study the problem of generating consecutive multiple future frames by observing one single still image only. We formulate the multi-frame prediction task as a multiple time step flow (multi-flow) prediction phase followed by a flow-to-frame synthesis phase. The multi-flow prediction is modeled in a variational probabilistic manner with spatial-temporal relationships learned through 3D convolutions. The flow-to-frame synthesis is modeled as a generative process in order to keep the predicted results lying closer to the manifold shape of real video sequence. Such a two-phase design prevents the model from directly looking at the high-dimensional pixel space of the frame sequence and is demonstrated to be more effective in predicting better and diverse results. Extensive experimental results on videos with different types of motion show that the proposed algorithm performs favorably against existing methods in terms of quality, diversity and human perceptual evaluation.

6.1 Introduction

Part of our visual world constantly experiences situations that require us to forecast what will happen over time by observing one still image from a single moment. Studies in neuroscience show that this *preplay* activity might constitute an automatic prediction mechanism in human visual cortex [30]. Given the great progress in artificial intelligence, researchers also begin to let machines learn to perform such a predictive activity for various applications. For example in Figure 6.1(top), from a snapshot by the surveillance camera, the system is expected to predict the man's next action which could be used for safety precautions. Another application in computational photography is turning still images into vivid cinemagraphs for aesthetic effects, as shown in Figure 6.1(bottom).

In this work, we mainly study how to generate pixel-level future frames in multiple time steps given one still image. A number of existing prediction models [83, 102, 118, 19] are under the assumption of observing a short video sequence ($>$1 frame). Since multiple historical frames explicitly exhibit obvious motion cues, most of them use deterministic models to render a fixed future sequence. In contrast, our single-image based prediction task, without any motion information provided, implies that there are obvious uncertainties existed in both spatial and temporal domains. Therefore we propose a probabilistic model based on a conditional variational autoencoder (cVAE) to model the uncertainty. Our probabilistic model has two unique features. First, it is a 3D-cVAE model, i.e., the autoencoder is designed in a spatial-temporal architecture with 3D convolution layers. The 3D convolutional layer [111], which takes a volume as input, is able to capture correlations between the spatial and temporal dimension of signals, thereby rendering distinctive spatial-temporal features for better predictions. Second, the output of our model is optical flows which characterize the spatial layout of how pixels are going to move step by step. Different from other methods that predict trajectories [125], frame differences [136] or frame pixels [19], the flow is a more natural and general representation of motions. It serves as a relatively low-dimensional reflection of high-level structures and can be ob-

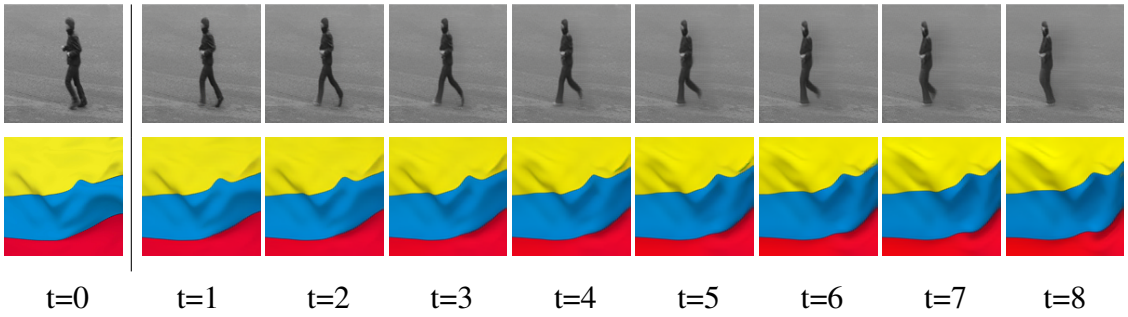


Figure 6.1: Multi-step future sequences generated by our algorithm (t=1∼8) conditioned on one single still image (t=0). Images are of size 128×128. For better view, each sequence shown in the work is animated as a video in the supplementary material.
tained in an unsupervised manner.

With the predicted flows, we next formulate the full frame synthesis as a generation problem. Due to the existence of occlusions, flow-based pixel-copying operations (e.g., warping) are obviously ineffective here. The model should be capable of "imagining" the appearance of future frames and removing the unnecessary parts in the previous frame at the same time. Therefore we propose a generative model *Flow2rgb* to generate pixel-level future frames. Such a model is non-trivial and is demonstrated to be effective in keeping the generated sequence staying close to the manifold of real sequences (Figure 6.5). Overall, we formulate the multi-frame prediction task as a multiple time step flow prediction phase followed by a flow-to-frame generation phase. Such a two-phase design prevents the model from directly looking at the high-dimensional pixel space of the frame sequence and is demonstrated to be more effective in predicting better results. During the testing, by drawing different samples from the learned latent distribution, our approach can also predict diverse future sequences.

The main contributions of this work are summarized as follows:

- We propose a spatial-temporal conditional VAE model (3D-cVAE) to predict future flows in multiple time steps. The diversity in predictions is realized by drawing different samples from the learned distribution.

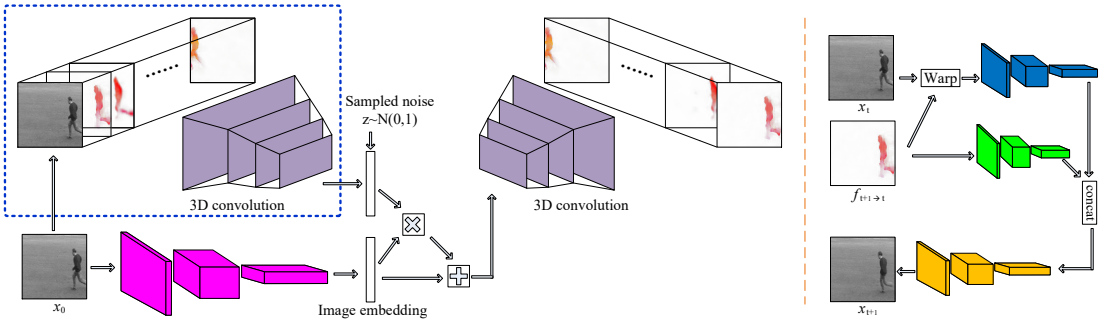


Figure 6.2: Architecture of the proposed multi-step prediction network. It consists of a 3D-cVAE (left) for predicting consecutive flows and a *Flow2rgb* model to generate future frame pixels (right). During the testing, the encoder (blue rectangle) of 3D-cVAE is no longer used and we directly sample points from the distribution for predictions.

- We present a generative model that learns to generate the pixel-level appearance of future frames based on predicted flows.
- We demonstrate the effectiveness of our method for predicting sequences that contain both articulated (e.g., humans) objects and dynamic textures (e.g., clouds).

6.2 Proposed Algorithm

We formulate the video prediction as two phases: flow prediction and flow-to-frame generation. The flow prediction phase, triggered by a noise, directly predicts a set of consecutive flow maps conditioned on the observed first frame. Then the flow-to-frame phase iteratively synthesizes future frames with the previous frame and the corresponding predicted flow map, starting from the first given frame and first predicted flow map.

6.2.1 Flow prediction

Figure 6.2(left) illustrates the architecture of our proposed model for predicting consecutive optical flows. Formally, our model is a conditional variational autoencoder [59, 107] with a spatial-temporal convolutional architecture (3D-cVAE). Given a sequence $X = \{x_i\}_0^M$ with x_0 as the starting frame, we denote the set of consecutive optical flows between adjacent frames in X as $F = \{f_i\}_0^{M-1}$. The network is trained to map the observation F (conditioned on x_0) to the latent variable z which are likely to reproduce the F. In order to avoid training a deterministic model, we produces a distribution over z values, which we sample from before the decoding. Such a variational distribution $q_\phi(z|x_0, F)$, known as the recognition model in [107], is assumed to be trained to follow a Gaussian distribution $p_z(z)$. Given a sampled z, the decoder decodes the flow F from the conditional distribution $p_\theta(F|x_0, z)$. Therefore the whole objective of network training is to maximize the variational lower-bound [59] of the following negative log-likelihood function:

$$\mathcal{L}(x_0, F; \theta, \phi) \approx -\mathcal{D}_{KL}(q_\phi(z|x_0, F)||p_z(z)) + \frac{1}{L}\sum_{1}^{L} \log p_\theta(F|x_0, z), \tag{6.1}$$

where $\mathcal{D}_{KL}$ is the Kullback-Leibler (K-L) divergence and L is the number of samples. Maximizing the term at rightmost in (6.1) is equivalent to minimizing the L1 distance between the predicted flow and the observed flow. Hence the loss $\mathcal{L}$ consists of a flow reconstruction loss and a K-L divergence loss.

Different from traditional cVAE models [107, 136, 125], our 3D-cVAE model employs the 3D convolution (purple blocks in Figure 6.2) which is demonstrated to be well-suited for spatial-temporal feature learning [111, 123]. In terms of network architecture, the 3D convolutional network outputs multiple (a volume of) flow maps instead of one, which can be used to predict multiple future frames. More importantly, the spatial-temporal relationship between adjacent flows are implicitly modeled during the training due to the 3D convolution operations, ensuring that the predicted motions are continuous and reasonable over time. In order to let the variational distribution $q_\phi(z|x_0, F)$ conditioned on

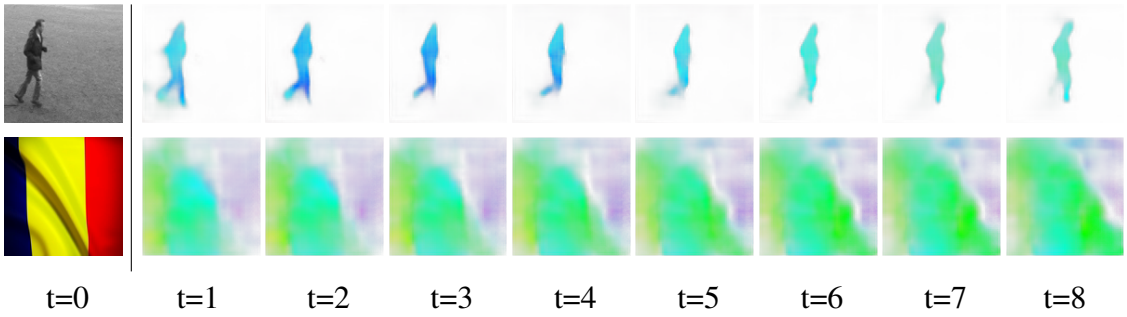


Figure 6.3: Examples of our multi-step flow prediction. During the testing, by simply sampling a noise from $N \sim (0, 1)$, we obtain a set of consecutive flows that describe the future motion field in multiple time steps. Note that since we have a warp operation in the later flow-to-frame step (Section 6.2.2) and the backward warping will not result in *holes* in results, we predict the backward flow in this step, i.e., the motion from x_{t+1} to x_t. This is just for convenience and we empirically do not find obvious difference between predicting forward and backward flows.

the starting frame, we stack x_0 with each flow map f_i in F as the encoder input. Meanwhile, learning the conditional distribution $p_\theta(F|x_0, z)$ for flow reconstruction also needs to be conditioned on x_0 in the latent space. Therefore, we propose an image encoder (pink blocks in Figure 6.2) to first map x_0 to a latent vector that has the same dimension as z. Inspired by the image analogy work [95], we use a conditioning strategy of combining the multiplication and addition operation, as shown in Figure 6.2(left). After we obtain the flow sequence for the future, we proceed to generate the pixel-level full frames.

6.2.2 Frame generation

Given the flow information, a common way to obtain the next frame is warping or pixel copying [152]. However, due to the existence of occlusions, the result is often left with unnecessary pixels inherited from the previous frame. The frame interpolation work [77] predicts a mask indicating where to copy pixels from previous and next frame. But they

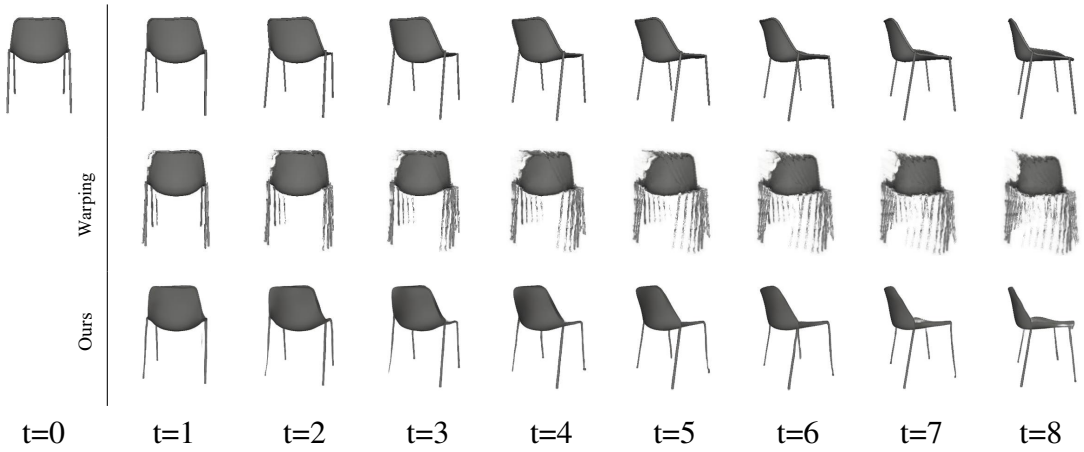
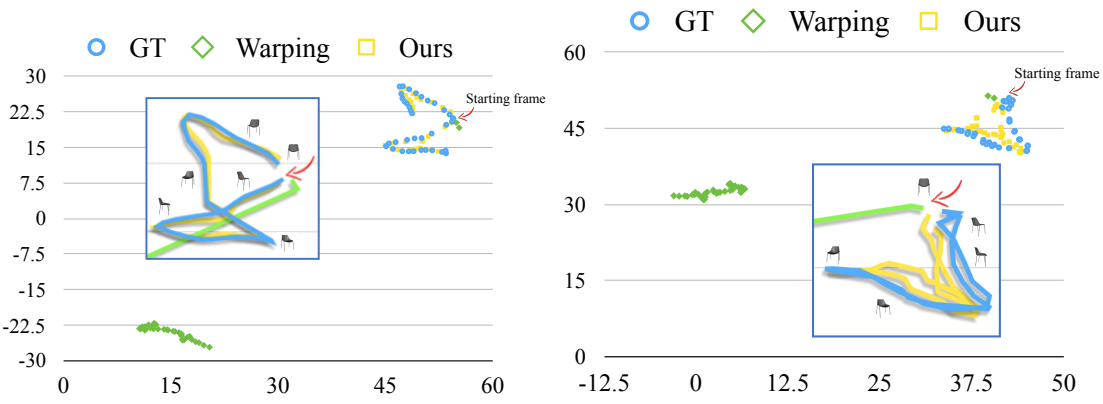


Figure 6.4: Comparisons between our *Flow2rgb* model and warping operation, given the first frame and all precomputed flows (between adjacent ground truth frames). Starting from the first frame and first flow, we iteratively run warping or the proposed *Flow2rgb* model based on the previous result and next flow to obtain the sequence. Top: ground truth, Middle: warping results, Bottom: our results.

require at least two frames to infer the occluded parts. Since we only observe one image, it is straightforward to formulate this step as a generation process, meaning that this model can "imagine" the appearance of next frame according to the flow and starting frame.

The architecture of the proposed frame generation model *Flow2rgb* is shown in Figure 6.2(right). Given the input x_t and its optical flow f_t that represents the motion of next time step, the network is trained to generate the next frame x_{t+1}. Since two adjacent frames often share similar information (especially in the static background regions), in order to let the network focus on learning the difference of two frames, we first warp the x_t based on the flow to get a coarse estimation $\tilde{x}_{t+1}$. Then we design a Siamese-like [15] network with the warped frame and the flow as two streams of input. The frame and flow encoders (blue and green blocks) borrow the same architecture of the VGG-19 up to the Relu_4_1 layer, and the decoder (yellow blocks) is designed as being symmetrical to the encoder with the nearest neighbor upsampling layer used for enlarging feature maps. We train the model



(a) VGG-19 pool5 (b) VGG-19 fc6

Figure 6.5: Visualization of sequence (a chair turning around) manifold in deep feature space. Staring from the same frame, each predicted frame of three sequences is visualized as a 2-D point by applying t-SNE [81] on its deep features. The moving average is shown as lines to imply the shape (or trending) of the manifold. For example in (a), the GT rotating chair (blue) follows a "8" like manifold in pool5 feature space, which our predicted sequence (yellow) follows closely but the warping sequence (green) deviates much further.

using a pixel reconstruction loss and a feature loss [56, 25] as shown below:

$$\mathcal{L} = \|\hat{x}_{t+1} - x_{t+1}\|_2 + \sum_{K=1}^{5} \lambda \|\Phi_K(\hat{x}_{t+1}) - \Phi_K(x_{t+1})\|_2 \,, \tag{6.2}$$

where $\hat{x}_{t+1}$, x_{t+1} are the network output and ground truth (GT), and Φ_K is the VGG-19 [105] encoder that extracts the Relu_K_1 features. λ is the weight to balance the two losses. This model is learned in an unsupervised manner without human labels. Note that this is a one-step flow-to-frame model. Since we predict multi-step flows in the flow prediction stage, starting with the first given frame, we iteratively run this model to generate the following frame based on the next flow and previous generated frame.

We show the effectiveness of our *Flow2rgb* model in Figure 6.4 with an example of chair rotating sequence [132]. To verify the frame generation phase alone, we assume that the flows are already available (computed by [94]). Then given the first frame and future

flows, the second row of Figure 6.4 shows the iterative warping results where the chair legs are repeatedly copied in future frames as the warping is unable to depict the right appearance of chair in difference views. In contrast, our model iteratively generates the occluded parts and removed unnecessary parts in the previous frame according to the flow at each time step. As claimed in [132], the deep embeddings of objects under consecutively changing views often follow certain manifold in feature space. If we interpret this changing view as a type of rotating motion, our predicted results for different views also needs to stay close to the manifold shape of the GT sequence. We demonstrate this by extracting the VGG-19 [105] features of each predicted frame, mapping it to a 2-D point through t-SNE [81], and visualizing it in Figure 6.5. It clearly shows that our predictions (in yellow) follows closely with the manifold of the GT sequence, while the warping drives the predictions to deviate from the GT further and further.

6.3 Experimental Results

In this section, we first discuss the experimental settings and implementation details. We then present qualitative and quantitative comparisons between the proposed algorithm and several competing algorithms. Finally, we analyze the diversity issue in uncertainty modeling. The source code will be made available to the public, and more results can be found in the supplementary material.

Datasets. We mainly evaluate our algorithm on three datasets. The first one is the KTH dataset [99] which is a human action video dataset that consists of six types of action and totally 600 videos. It represents the movement of articulated objects. Same as in [118, 19], we use person 1-16 for training and 17-25 for testing. We also collect another two datasets from online websites, i.e., the *WavingFlag* and *FloatingCloud*. These two datasets represents dynamic texture videos where motions may bring the shape changes on dynamic patterns. The *WavingFlag* dataset contains 341 videos of 80K+ frames and the *Floating-*

Cloud dataset has 415 videos of 150K+ frames in total. In each dataset, we randomly split all videos into the training (4/5) and testing (1/5) set. We do not take the camera motion into considerations and thus all videos in three datasets are stabilized.

Implementation details. Given the starting frame x_0, our algorithm predicts the future in next $M = 16$ time steps. Each frame is resized to 128×128 in experiments. Similar to [126, 34], we employ an existing optical flow estimator SPyNet [94] to obtain flows between GT frames for training the 3D-cVAE. As described in Section 6.2.1, we stack x_0 with each flow map f_i in F. Thus during the training, the input cube to the 3D-cVAE is of size $16 \times 5 \times 128 \times 128$ where $5 = 2 + 3$ (2-channel flow and 3-channel RGB). The dimension of the latent variable z in the bottle neck is set as 2000. The detailed network architecture can be found in the supplementary material. Another important factor for a successful network training is to normalize the flow roughly to (0,1) before feeding it into the network, ensuring pixel values of both flows and RGB frames are within the similar range. Since the *Flow2rgb* model can be an independent module for motion transfer with known flows, we train the 3D-cVAE and *Flow2rgb* model separately in experiments.

Evaluations. Different prediction algorithms have their unique settings and assumptions. For example, Mathieu *et al.* [83] requires four frames stacked together as the input. Villegas *et al.* [118] ask for feeding the image difference (at least two frames). Their following work [119], though based on one frame, additionally needs multiple historical human pose maps to start the prediction. For fair comparisons, we mainly select prediction methods [19, 136] that accept one single image as the only input to compare. The work of [19] represents the typical recursive prediction pipeline, which builds upon a fully-connected long short-term memory (FC-LSTM) layer for predictions. Their model is originally trained and tested by observing multiple frames. Here we change their setting to one-frame observance in order to be consistent with our setting. The work of [136] is the typical one-step prediction method based on one given frame. To get multi-frame

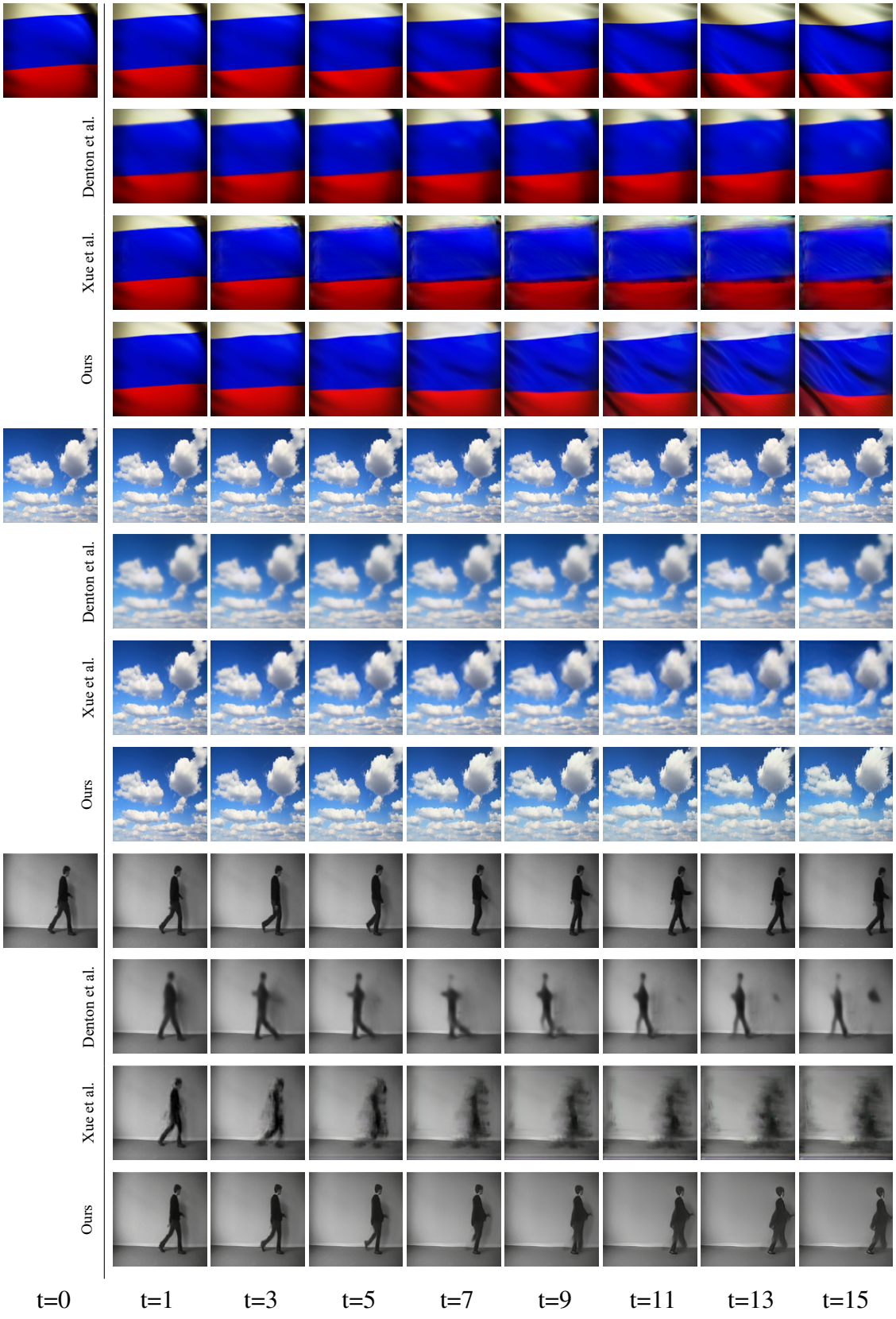
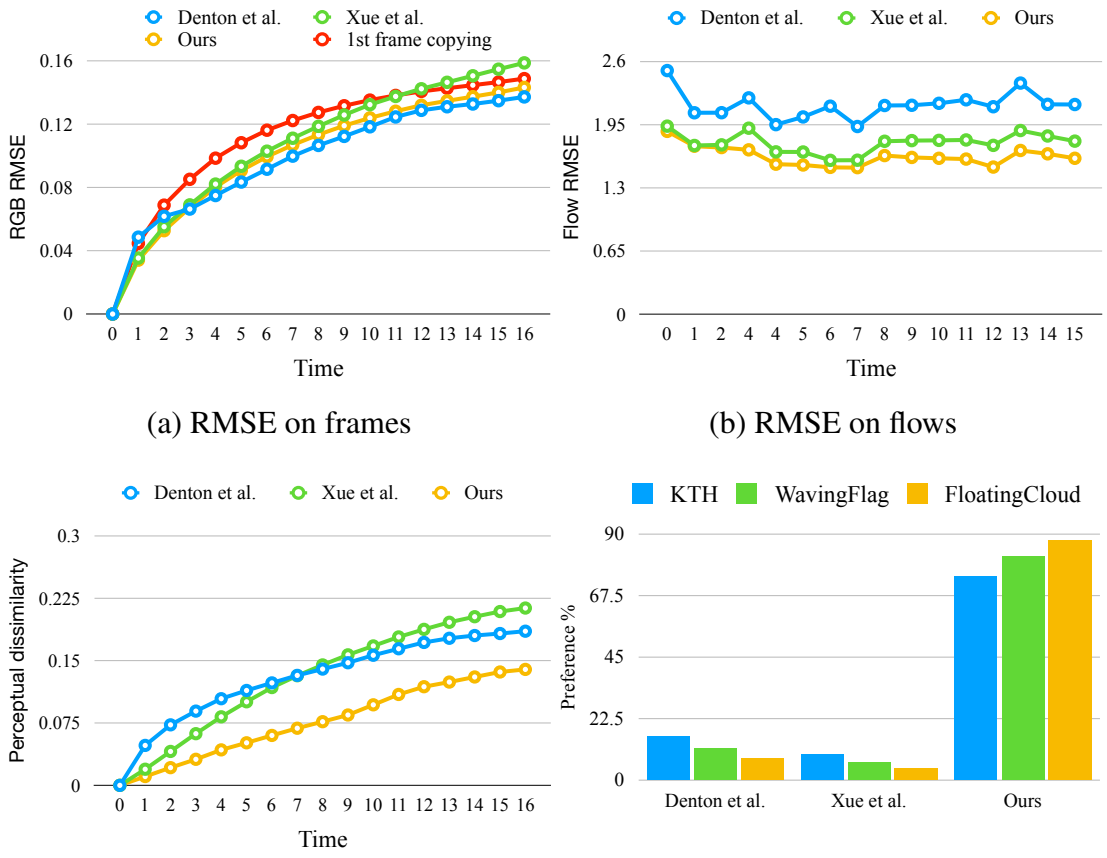


Figure 6.6: Visual comparisons of different prediction algorithms. Top left: the starting frame. From top to bottom in example: GT, Denton *et al.* [19], Xue *et al.* [136], Ours.

predictions, we train their model and iteratively test it to get the next prediction based on the previous prediction.

In Figure 6.6, we provide a visual comparison between the proposed algorithm and [19, 136]. In [19], a pre-trained and disentangled *pose* embedding is employed to keep predicting the pose of the next frame through a FC-LSTM module. For articulated objects, the pose is often compact and in low dimensions, which is relatively easier to handle with a single LSTM module. However, for dynamic textures (e.g., flag, cloud) where all pixels are likely to move, the global pose becomes complex and is no longer a low-dimensional structure representation. Therefore the capacity of recursive models is not enough to capture the spatial and temporal variation trend at the same time. The first two examples in Figure 6.6 show that the flag and cloud in predicted frames are nearly static. Meanwhile, the pose only describes the static structure of the object in the current frame and cannot tell as much information as the flow about the next-step motion. In the third example of Figure 6.6, it is obvious that the human is walking to the right. But the results of [19] show that the human is going in a reverse direction. Moreover, since they directly predict frame pixels and use the reconstruction loss only, their results are relatively blurry. In [136], as they only predict the next one frame, the motion is often clear in the second frame. But after we keep predicting the following frame using the previous predicted frame, the motion gradually disappears and the quality of results degrades fast during a few steps. Moreover, they choose to predict the image difference which only shows global image changes but does not capture how each pixel will move to its corresponding one in the next frame. In contrast, our results show more continuous and reasonable motion, reflected by better generated full frames. For example, in the first flag example, the starting frame indicates that the fold on top right will disappear and the fold at bottom left will bring bigger folds. Our predicted sequence presents the similar dynamics as what happens in the GT sequence, which makes it look more realistic. Please see the animated results in the supplementary material for better view.



(a) RMSE on frames

(b) RMSE on flows

(c) Perceptual metric [149] on frames

(d) Which sequence looks more realistic?

Figure 6.7: Quantitative evaluations of different prediction algorithms. We start from the per-pixel metrics (e.g., RMSE) and gradually take human perception into consideration. Our results are demonstrated to be visually more similar to the GT sequence and achieves the best performance under metrics (b)-(d).

We also quantitatively evaluate these prediction methods using three different metrics, i.e., the root-mean-square error (RMSE), perceptual similarity [149], and user preference. The RMSE is the classic per-pixel metric which measures the spatial correspondence without considering any high-level semantics and is often easily favored by smooth results. Based on this observation, the recent work of [149] proposes a perceptual similarity metric by using deep network embeddings. It is demonstrated to agree with human percep-

tions better. Lastly, we directly ask the feedback from users by conducting user studies to understand their preference towards the predicted results by different algorithms.

We start with the traditional RMSE to compute the difference between predicted sequence and GT sequence frame-by-frame and show the result in Figure 6.7(a). To understand how effective these prediction methods are, we design a simple baseline by copying the given frame as multi-step predictions. However, we do not observe obvious difference among all these methods. While the prediction from one single image is originally ambiguous, the GT sequence can be regarded as just one possibility of the future. The trending of motion may be similar but the resulted images can be significantly different in pixel-level. But the RMSE metric is actually very sensitive to the pixel spatial mismatch. Similar observations are also found in [19, 149]. That is why all these methods, when comparing with the GT sequence, shows the similar RMSE results. Therefore, instead of measuring the RMSE on frames, we turn to measure the RMSE on optical flows because the optical flow represents whether the motion field is predicted similarly or not. Here we do not include the naive copying approach as there are no flows in its sequence. We compute the flow maps between adjacent frames of the GT sequence and other predicted sequences using the SPyNet [94] and show the RMSE results in Figure 6.7(b). Now the difference becomes more clear and our method achieves the lowest RMSE results, meaning that our prediction is the closest to the GT in terms of the predicted motions.

However, the evaluation of prediction results still need to take human perception into consideration in order to determine whether sequences look as realistic as the GT sequence. Therefore we turn to the perceptual similarity metric [149]. We use the AlexNet [63] for feature extraction and measure the similarity between predicted sequence and GT sequence frame-by-frame. Since this metric is obtained by computing feature distances, we denote it as perceptual dissimilarity so that small values means being more similar. The results in Figure 6.7(c) show that the proposed method outperforms other algorithms with an even larger margin than that in Figure 6.7(b), which means that the predicted sequence of our method is perceptually more similar to the GT sequence.

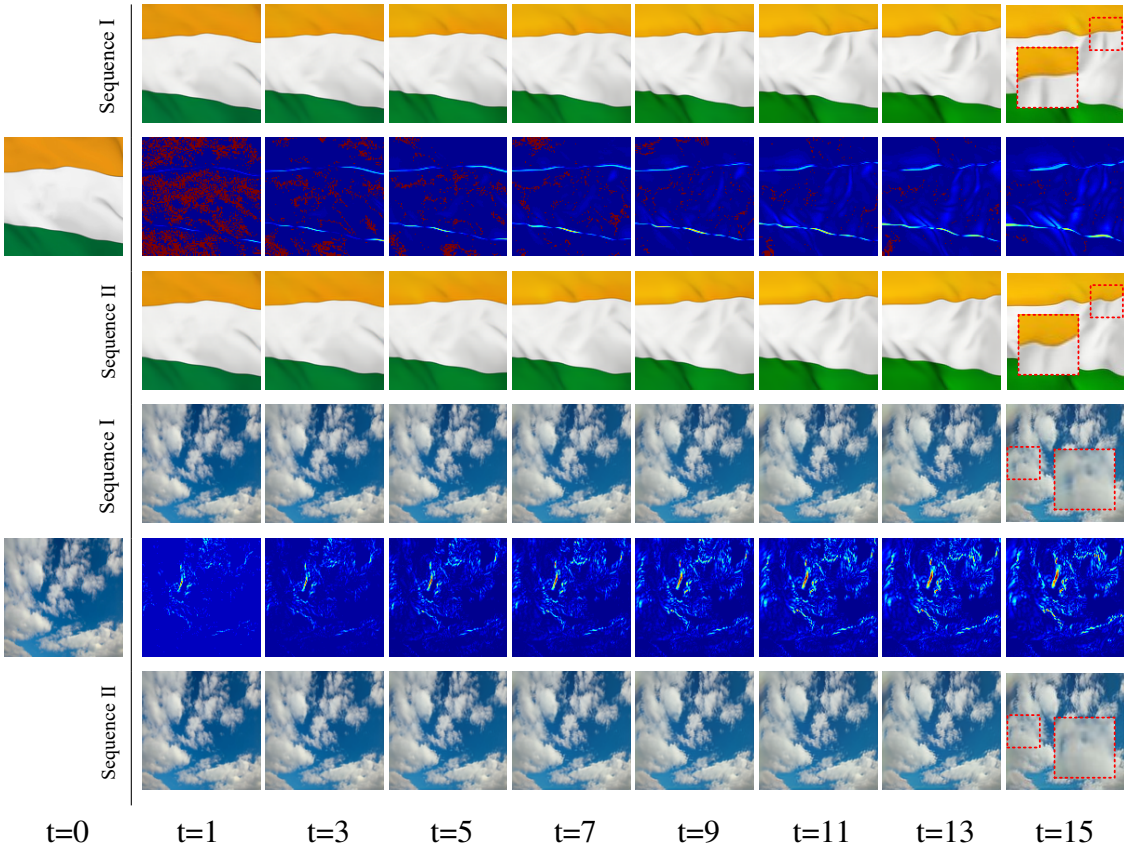
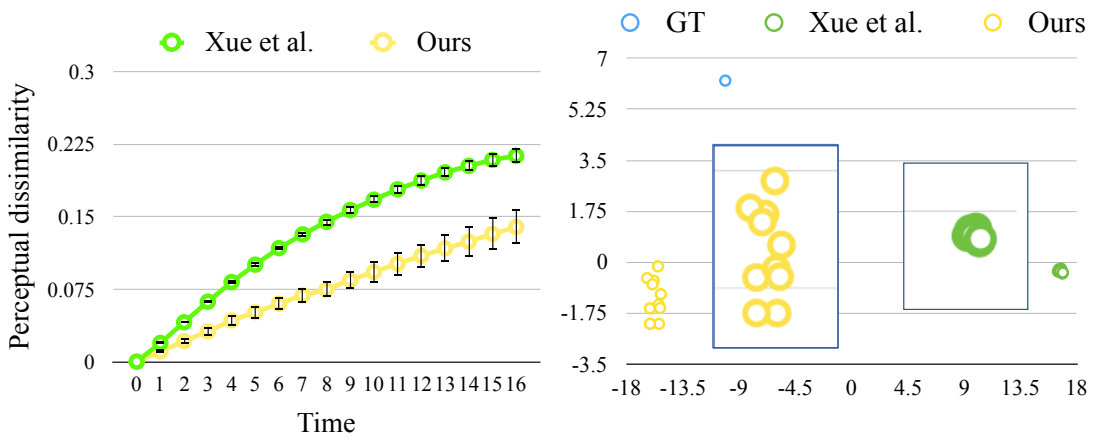


Figure 6.8: Given a still image, by sampling different noise in the latent space, our algorithm synthesizes different future outcomes (top and bottom) to account for the intrinsic uncertainties. In the middle row, we show the difference of two generated sequences frame-by-frame. Note that the diversity is not reflected by pixel intensities. The difference sequence shows that the shape of flag and cloud changes in different way. We highlight two regions in the last column.

Finally, we conduct the user study to get the feedback from human subjects on judging different predicted results. We prepare 30 starting frames (10 from each dataset) and generated 30 sequences (16-frame) for each method. For each subject, we randomly select 15 sets of sequences predicted by three methods. For each starting frame, the three predicted sequences are displayed side-by-side in random order. Here we do not include the



(a) Sensitivity on the perceptual quality under different noise

(b) Visualized distribution of predictions under different noise

Figure 6.9: Comparisons between [136] and the proposed algorithm on uncertainty modeling given the same starting frame. By drawing different samples, the generated predictions by our method exhibits more diversities while still being more similar (closer) to the GT sequence. In (b), we zoom in the distribution of different predicted sequences in the middle to show the diversity clearly.

naive copying approach as the static prediction is not assumed to be a kind of future in this work. Each subject is asked to vote one sequence that looks most realistic for each starting frame. We finally collect 900 votes from 60 users and report the results (in percentage) in Figure 6.7(d). The study results clearly show that the proposed method receives the most votes for more realistic predictions among all three categories. Both Figure 6.7(c) and (d) indicate that the proposed method performs favorably against [19, 136] in terms of perceptual quality.

Diversity. Both [136] and the proposed method model the uncertainty in predictions, but are different in one-step [136] or multi-step uncertainties. By drawing different samples, we evaluate how the quality of predictions is affected by the noise input and how diverse

the predicted sequences are. While [136] uses a noise vector of 3200 dimensions and we use that of 2000 dimensions, the noise inputs of two models are not exactly the same but they are all sampled from $N(0, 1)$. We sample 10 noise inputs for each method, while ensuring that the two sets of noise inputs have the similar mean and standard deviation. Then we obtain 10 sequences for each method, and compare them with the GT sequence. Figure 6.9(a) shows the mean and standard deviation of the perceptual metric over each method's 10 predictions when compared with the GT frame-by-frame. Under different noise inputs, our method keeps generating better sequences that are more similar to the GT. Meanwhile, the results of our algorithm show larger deviation, which implies that there are more diversities in our predictions. To further verify this, we show the embeddings of generated sequences in Figure 6.9(b). For each sequence, we extract the VGG-19 [105] features (e.g., fc6 layer) of each frame, stack them as one vector, and map it to a 2-D point through t-SNE [81]. Figure 6.9(b) shows that our 10 predictions are much closer to the GT sequence while being scattered to be different from each other. In contrast, the 10 predictions of [136] huddle together and are far from the GT. The findings in Figure 6.9(a) and (b) consistently tell that the proposed algorithm generates more realistic and diverse future predictions.

Bringing still images to life. Unlike previous video prediction methods [118, 119, 125] that mainly focus on humans for action recognition, our algorithm is more general towards bringing elements in the still image to life, i.e., turning a still image into a vivid GIF for aesthetic effects. It can be an effective tool for video editing.

In Figure 6.10(a), we show a example of turning a photo into a vivid sequence. We mask out the sky region, apply our model trained on the *FloatingCloud* dataset and generate the effect of clouds floating in the sky. This could further benefit existing sky editing methods [112]. Moreover, if we replace our flow prediction with known flows from a reference sequence, our flow-to-frame model *Flow2rgb* becomes a global motion style transfer model. Figure 6.10(b) shows such an example of transferring the referenced mo-

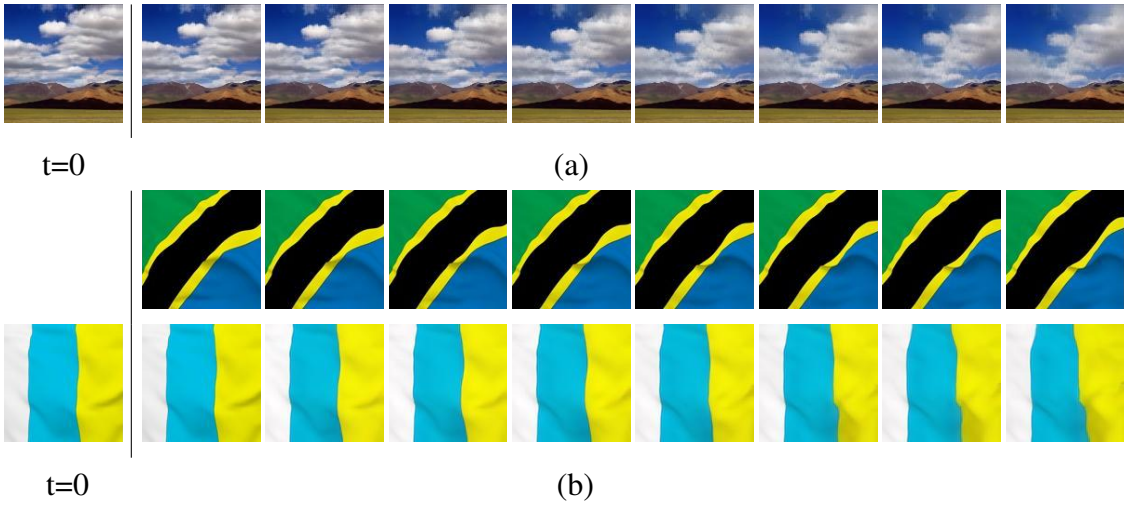


Figure 6.10: Potential application of our algorithm in video editing. (a) Motion prediction and frame generation. (b) Motion transfer from a reference sequence.

tion (top) of a waving flag to another still flag image (bottom left). We recommend the readers to look at these two video applications in the supplementary materials. As the current random sampling strategy for flow predictions is uncontrollable, future work may include introducing more interactions from users to control detailed motions.

6.4 Conclusions

In this work, we propose a video prediction algorithm that synthesizes a set of likely future frames in multiple time steps from one single still image. Instead of directly estimating the high-dimensional future frame space, we choose to decompose this task into a flow prediction phase and a flow-grounded frame generation phase. The flow prediction models the future uncertainty and spatial-temporal relationship in a 3D-cVAE model. The frame generation step helps prevent the manifold shape of predicted sequences from straying off the manifold of real sequences. We demonstrate the effectiveness of the proposed algorithm on both human action videos and dynamic texture videos.

Chapter 7

Conclusion and Future Work

7.1 Summary

Regarding this topic of visual synthesis, we have investigated four different tasks, including structure enhancement, style transfer, content filling and motion prediction. We have made several key contributions towards constructing effective computational models to preserve visual realism and facilitate more stunning creations.

Chapter 3 presents a learning-based approach for joint filtering based on convolutional neural networks. Instead of relying only on the guidance image, we design two sub-networks to extract informative features from both the target and guidance images. We show that the proposed algorithm is computationally efficient and performs favorably against the state-of-the-art techniques on a wide variety of computer vision and computational photography applications, including cross-modal denoising, joint image upsampling, and texture-structure separation.

Chapter 4 presents a universal style transfer algorithm that does not require learning for each individual style. By unfolding the image generation process via training an auto-encoder for image reconstruction, we integrate the whitening and coloring transforms in the feed-forward passes to match the statistical distributions and correlations between the

intermediate features of content and style. Experimental results demonstrate that the proposed algorithm achieves favorable performance against the state-of-the-art methods in generalizing to arbitrary artistic and photo styles.

In Chapter 5, we propose a deep generative network for face completion. The network is based on a GAN, with an autoencoder as the generator, two adversarial loss functions (local and global) and a semantic regularization as the discriminators. The proposed model can successfully synthesize semantically valid and visually plausible contents for the missing facial key parts from random noise. Both qualitative and quantitative experiments show that our model generates the completion results of high perceptual quality and is quite flexible to handle a variety of maskings or occlusions.

Finally in Chapter 6, we present a video prediction algorithm that synthesizes a set of likely future frames in multiple time steps from one single still image. Instead of directly estimating the high-dimensional future frame space, we choose to decompose this task into a flow prediction phase and a flow-grounded frame generation phase. The flow prediction models the future uncertainty and spatial-temporal relationship in a 3D-cVAE model. The frame generation step helps prevent the manifold shape of predicted sequences from straying off the manifold of real sequences. We demonstrate the effectiveness of the proposed algorithm on both human action videos and dynamic texture videos.

7.2 Future Work

The task of synthesis is far from being solved. Even for just one visual factor, it contains different levels of manipulation. In addition, there is still a gap between the computer-generated results and creations from artists or designers. A more general question is that whether the computer can create art. Based on the current achievements, we present three directions for future investigation.

Figure 7.1: The more challenging non-texture style to transfer.

7.2.1 Non-Texture Style Transfer

Manipulating the style is the core part of this thesis. While all proposed methods create amazing results, they are still only about the "texture". However, the style of an image can be reflected in different ways. As shown in Figure 7.1(a)-(b), the style is mostly about shape in face caricatures while in drawings the style focuses more on strokes or brushes. The concept of style is even hard to articulate when it comes to connect with the artist himself (Figure 7.1(c)). It is relatively unlikely to expect using just one model to learn or transfer those challenging styles. If we carefully observe the creation process of an artist or a painter, their drawing usually have clear steps, such as abstraction, generalization, exaggeration, and details. A better way is to use the procedural modeling to approximate the human behavior instead of finishing everything in one step with computers. Another long-term goal of this direction is to provide more fine-grained control to neural-network stylization algorithms. Existing learning-based methods do not provide much control over style except by changing the training input. We desire to facilitate users' creations rather than totally separate them from being involved.

7.2.2 Disentangling Visual Factors

Though we successfully realize to manipulate four visual factors (i.e., structure, style, content and motion) for the synthesis, we do not truly disentangle them from the observed data. Learning disentangled representations of those factors are important to understand how visual data is formed and can be generated. In addition, it is also useful for cross-domain scenarios if we could figure out the factors that are either domain-invariant or domain-variant. What the existing synthesis techniques could contribute is that we are able to synthesize large amounts of new data that are very likely to serve as the training data for learning disentangled representations. For example, it is unlikely to collect thousands of Picasso artworks but we can use the universal style transfer technique to turn any photo into Picasso-style version. Some recent work [19] already start exploring this direction with the adversarial learning.

7.2.3 Unsupervised Representation Learning via Visual Synthesis

In the classical supervised tasks (e.g., classification), it usually requires a large amount of labeled data and is thus labor intensive. It is often expected to learn unsupervised visual representations [21, 89, 121] with large amount of unlabeled data that could help boost the supervised performance. The visual synthesis, which is usually self-supervised, exactly provides the possibility of learning good representations by synthesizing realistic results. Take the content filling as example. It is argued that doing well on this task requires the model to learn powerful representations about objects and their parts, which can be then transferred to related supervised tasks (e.g., object classification). Given a large, unlabeled image collection, we randomly remove several regions from objects in each image and train a CNN to fill in the missing regions. The input image itself serves as the label to facilitate the training. Especially for completing the objects rather than the background, this will greatly increase the difficulty of network learning but only in this way will the network be possibly pushed to learn more effective features. While the current

performance of unsupervised representation learning is still far behind that of supervised learning, we have a long way to go exploring different kinds of synthesis tasks that could contribute to the general representation learning.

Bibliography

[1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009.

[2] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[3] V. Andrea and L. Karel. MatConvNet – convolutional neural networks for matlab. In *ACM MM*, 2015.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28(3):24, 2009.

[5] J. T. Barron and B. Poole. The fast bilateral solver. In *ECCV*, 2016.

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH*, 2000.

[8] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *TIP*, 12(8):882–889, 2003.

[9] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *CVPR*, 2012.

[10] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[11] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016.

[12] Y.-W. Chao, J. Yang, B. Price, S. Cohen, and J. Deng. Forecasting human dynamics from static images. In *CVPR*, 2017.

[13] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017.

[14] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

[15] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[16] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[17] F. Cutzu, R. Hammoud, and A. Leykin. Estimating the photorealism of images: Distinguishing paintings from photographs. In *CVPR*, 2003.

[18] E. David, P. Christian, and F. Rob. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[19] E. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017.

[20] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *NIPS*, 2005.

[21] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[22] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[23] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

[24] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *PAMI*, 38(2):295 – 307, 2015.

[25] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016.

[26] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016.

[27] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

[28] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013.

[29] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. In *SIGGRAPH*, 2004.

[30] M. Ekman, P. Kok, and F. P. de Lange. Time-compressed preplay of anticipated events in human primary visual cortex. *Nature Communications*, 8, 2017.

[31] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *ICCV*, 2013.

[32] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.

[33] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In *CVPR*, 2016.

[34] R. Gao, B. Xiong, and K. Grauman. Im2flow: Motion hallucination from static images for action recognition. *arXiv preprint arXiv:1712.04109*, 2017.

[35] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[36] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[37] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. *arXiv preprint arXiv:1611.07865*, 2016.

[38] P. Georg, A. Maneesh, H. Hugues, S. Richard, C. Michael, and T. Kentaro. Digital photography with flash and no-flash image pairs. In *SIGGRAPH*, 2004.

[39] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand. Deep bilateral learning for real-time image enhancement. In *SIGGRAPH*, 2017.

[40] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017.

[41] R. C. Gonzalez and R. E. Woods. Digital image processing (3rd edition). *Prentice Hall*, 2008.

[42] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[43] S. Gu, W. Zuo, S. Guo, Y. Chen, C. Chen, and L. Zhang. Learning dynamic guidance for depth image enhancement. In *CVPR*, 2017.

[44] B. Ham, M. Cho, and J. Ponce. Robust image filtering using joint static and dynamic guidance. In *CVPR*, 2015.

[45] J. Hays and A. A. Efros. Scene completion using millions of phoraphs. *ACM Transactions on Graphics*, 26(3):4, 2007.

[46] K. He, J. Sun, and X. Tang. Guided image filtering. *PAMI*, 35(6):1397–1409, 2013.

[47] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[48] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*, 2001.

[49] H. Hirschmüller and D. Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, 2007.

[50] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 107(2):191–202, 2014.

[51] M. Hossain. Whitening and coloring transforms for multivariate gaussian random variables. *Project Rhea*, 2016.

[52] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM Transactions on Graphics*, 33(4):129, 2014.

[53] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *arXiv preprint arXiv:1703.06868*, 2017.

[54] T.-W. Hui, C. C. Loy, and X. Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, 2016.

[55] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *CVPR*, 2016.

[56] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[57] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing image style. In *BMVC*, 2014.

[58] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[59] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[60] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, pages 201–214, 2012.

[61] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. In *SIGGRAPH*, 2007.

[62] J. Kopf and D. Lischinski. Digital reconstruction of halftoned color comics. In *SIGGRAPH*, 2012.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[64] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014.

[65] A. Larsen, S. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

[66] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang. Interactive facial feature localization. In *ECCV*, 2012.

[67] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *SIGGRAPH*, 2004.

[68] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *PAMI*, 30(2):228–242, 2008.

[69] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016.

[70] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, 2016.

[71] X. Liang, L. Lee, W. Dai, and E. P. Xing. Dual motion gan for future-flow embedded video prediction. In *ICCV*, 2017.

[72] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[73] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[74] M.-Y. Liu, O. Tuzel, and Y. Taguchi. Joint geodesic upsampling of depth images. In *CVPR*, 2013.

[75] S. Liu, J. Yang, C. Huang, and M.-H. Yang. Multi-objective convolutional learning for face labeling. In *CVPR*, 2015.

[76] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[77] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017.

[78] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[79] S. Lu, X. Ren, and F. Liu. Depth enhancement via low-rank matrix completion. In *CVPR*, 2014.

[80] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *CVPR*, 2017.

[81] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

[82] L. Mai and F. Liu. Comparing salient object detection results without ground truth. In *ECCV*, 2014.

[83] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[84] R. Mechrez, E. Shechtman, and L. Zelnik-Manor. Photorealistic style transfer with screened poisson equation. In *BMVC*, 2017.

[85] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[86] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[87] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, 2015.

[88] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. High quality depth map upsampling for 3d-tof cameras. In *ICCV*, 2011.

[89] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[90] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *SIGGRAPH*, 2003.

[91] F. Philipp, D. Alexey, I. Eddy, H. Philip, H. Caner, G. Vladimir, V. d. S. Patrick, C. Daniel, and B. Thomas. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[92] F. Pitié, A. C. Kokaram, and R. Dahyot. N-dimensional probability density function transfer and its application to color transfer. In *ICCV*, 2005.

[93] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[94] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017.

[95] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In *NIPS*, 2015.

[96] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

[97] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *NIPS*, 2015.

[98] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, 2007.

[99] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

[100] X. Shen, C. Zhou, L. Xu, and J. Jia. Mutual-structure for joint filtering. In *ICCV*, 2015.

[101] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[102] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.

[103] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. Style transfer for headshot portraits. In *SIGGRAPH*, 2014.

[104] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. In *SIGGRAPH*, 2013.

[105] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[106] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang. Exemplar-based face parsing. In *CVPR*, 2013.

[107] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[108] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015.

[109] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[110] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[111] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[112] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, and M.-H. Yang. Sky is not the limit: Semantic-aware sky replacement. *ACM Transactions on Graphics*, 35(4):149, 2016.

[113] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. *arXiv preprint arXiv:1707.04993*, 2017.

[114] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[115] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[116] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017.

[117] A. Van den Oord, N. Nal Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[118] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017.

[119] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*, 2017.

[120] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[121] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

[122] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016.

[123] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, 2016.

[124] C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *CVPR*, 2017.

[125] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016.

[126] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *ICCV*, 2015.

[127] H. Wang, X. Liang, H. Zhang, D.-Y. Yeung, and E. P. Xing. Zm-net: Real-time zero-shot image manipulation network. *arXiv preprint arXiv:1703.07255*, 2017.

[128] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. *arXiv preprint arXiv:1603.05631*, 2016.

[129] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *CVPR*, 2017.

[130] P. Wilmot, E. Risser, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[131] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *PAMI*, 31(2):210–227, 2009.

[132] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[133] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[134] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, 2015.

[135] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics*, 31(6):139, 2012.

[136] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016.

[137] Q. Yan, X. Shen, L. Xu, S. Zhuo, X. Zhang, L. Shen, and J. Jia. Cross-field joint image restoration via scale map. In *ICCV*, 2013.

[138] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013.

[139] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, 2016.

[140] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolution for range images. In *CVPR*, 2007.

[141] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017.

[142] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017.

[143] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep joint image filtering. In *ECCV*, 2016.

[144] J. Yuen and A. Torralba. A data-driven approach for event prediction. In *ECCV*, 2010.

[145] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[146] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2005.

[147] J. Zhang, J. Pan, W.-S. Lai, R. Lau, and M.-H. Yang. Learning fully convolutional networks for iterative non-blind deconvolution. In *CVPR*, 2017.

[148] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, 2014.

[149] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep networks as a perceptual metric. In *CVPR*, 2018.

[150] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun. Stacked what-where auto-encoders. In *ICLR Workshop*, 2016.

[151] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *NIPS*, 2004.

[152] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *ECCV*, 2016.

[153] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, pages 479–486, 2011.